fuddly Documentation

Release 0.30

Eric Lacombe

Mar 01, 2023

CONTENTS

1 Overview **3**

2 Tutorial **5**

2.1 Using `fuddly` simple UI: `Fuddly Shell` . . . 5

2.1.1 Start a Fuzzing Session . . . 5

2.1.2 Send Malformed ZIP Files to the Target (Manually) . . . 8

2.1.2.1 How to Send a ZIP File . . . 8

2.1.2.2 How to Perform Automatic Modification on Data . . . 10

2.1.2.3 Resetting & Cloning Disruptors . . . 17

2.1.2.4 Reloading Data Models / Targets / . . . 17

2.1.3 Use an Operator to Send Malformed Data . . . 18

2.1.4 Replay Data From a Previous Session . . . 19

2.2 Using `fuddly` Through Advanced Python Interpreter . . . 20

2.3 Implementing a Data Model and Defining a Project Environment . . . 23

2.3.1 Data Modeling . . . 23

2.3.1.1 Overview . . . 23

2.3.1.2 A First Example . . . 27

2.3.1.3 Defining the Imaginary MyDF Data Model . . . 28

2.3.2 Visualization of Modeled Data . . . 32

2.3.3 Absorption of Raw Data that Complies to the Data Model . . . 32

2.3.3.1 A First Example . . . 32

2.3.3.2 Absorption Constraints . . . 34

2.3.3.3 Defining Absorption Helpers . . . 35

2.3.4 Describing Protocols Ruling a Data Model . . . 36

2.3.5 Defining Specific Disruptors . . . 37

2.3.5.1 Overview . . . 37

2.3.5.2 The Model Walker Infrastructure . . . 38

2.3.6 Defining a Project Environment . . . 40

2.3.6.1 Defining the Targets . . . 41

2.3.6.2 Defining the Logger . . . 42

2.3.6.3 Defining Operators . . . 43

2.3.6.4 Defining Probes . . . 45

2.3.6.5 Defining Tasks . . . 48

3 Data Modeling **49**

3.1 Data Model Keywords . . . 49

3.1.1 Generic Description Keywords . . . 49

3.1.2 Keywords to Describe Non Terminal Node . . . 53

3.1.3 Keywords to Describe Generator Node . . . 55

3.1.4 Keywords to Import External Data Description . . . 55

3.1.5 Keywords to Describe Node Properties . . . 55
3.1.6 Keywords to Describe Constraints . . . 58
3.2 Value Types . . . 58
3.2.1 Integer . . . 59
3.2.2 String . . . 60
3.2.3 BitField . . . 61
3.3 Helpers . . . 63
3.3.1 Generator Node Templates . . . 63
3.3.2 Block Builders . . . 64
3.4 Data Model Patterns . . . 66
3.4.1 How to Describe Different Shapes for Some Parts of Data . . . 66
3.4.2 How to Describe the Separators of a Data Format . . . 68
3.4.3 How to Describe a Data Format Whose Parts Change Depending on Some Fields . . . 69
3.4.4 How to Generate Nodes Dynamically (for length, counter, ...) . . . 71
3.4.5 How to Describe a Data Format With Some Encoded Parts . . . 75
3.4.6 How to Describe a Data Format That Contains Complex Strings . . . 76
3.4.7 How to Describe Constraints of Data Formats . . . 78

4 Data Manipulation **81**
4.1 Overview . . . 81
4.1.1 Generate Data a.k.a. Freeze a Graph . . . 83
4.1.2 Create Nodes with Low-Level Primitives . . . 83
4.1.3 Cloning a Node . . . 84
4.1.4 Display a Frozen Graph . . . 84
4.1.5 The Node Environment . . . 85
4.2 Search for Nodes in a Graph . . . 85
4.3 The Node Dictionary Interface . . . 87
4.4 Change a Node . . . 88
4.4.1 Operations on Node Properties and Attributes . . . 90
4.4.2 Node Configurations . . . 91
4.4.3 Node Corruption Infrastructure . . . 92
4.4.4 Byte String Absorption . . . 93
4.5 Miscellaneous Primitives . . . 93
4.6 Entangled Nodes . . . 93

5 Data Analysis **95**
5.1 FmkDB Toolkit . . . 95
5.1.1 Usage Examples . . . 95
5.1.2 Fmkdb Toolkit Manual . . . 95
5.2 Plotty . . . 98
5.2.1 Usage Examples . . . 98
5.2.2 Plotty Manual . . . 98
5.2.3 Concrete output example . . . 99

6 Scenario Infrastructure **101**
6.1 Overview . . . 101
6.2 Scenario Description . . . 101
6.2.1 A First Example . . . 101
6.2.2 Steps . . . 106
6.2.3 Transitions . . . 107
6.2.4 Data Generation Process . . . 110
6.2.5 Scenario Involving Multiple Targets . . . 111
6.3 Scenario Fuzzing . . . 111
6.3.1 Overview . . . 111

6.3.2 Fuzzing by Example . . . 112
6.3.2.1 Invert Transition Conditions . . . 113
6.3.2.2 Ignore Timing Constraints . . . 116
6.3.2.3 Fuzz the Data Sent by the Scenario . . . 116
6.3.2.4 Make the protocol stutter . . . 117

7 Knowledge Infrastructure **119**
7.1 Get Knowledge about the Targets Under Test . . . 119
7.1.1 Defining a Feedback Handler to Create Knowledge from Targets' and Probes' Feedback . . . 119
7.1.2 Adding Knowledge About the Targets Under Test in the Project File . . . 121
7.1.3 Information Categories and How to Define More . . . 121
7.2 Leveraging the Knowledge . . . 121
7.2.1 Automatic Fuddly Adaptation to Knowledge . . . 121
7.2.2 Leveraging Knowledge in User-defined Components . . . 122

8 Evolutionary Fuzzing **123**
8.1 Overview . . . 123
8.2 User interface . . . 125
8.3 Crossover Algorithms . . . 126
8.3.1 Algo1 - Randomly swap some root nodes' children . . . 126
8.3.2 Algo2 - Randomly swap some leaf nodes . . . 126

9 Generic Data Makers **129**
9.1 Generic Generators . . . 129
9.1.1 GENP - Pattern Generation . . . 129
9.1.2 POPULATION - Generator for Evolutionary Fuzzing . . . 129
9.2 Generic Disruptors . . . 130
9.2.1 Stateful Disruptors . . . 130
9.2.1.1 tTYPE - Advanced Alteration of Terminal Typed Node . . . 130
9.2.1.2 tSTRUCT - Alter Data Structure . . . 132
9.2.1.3 tALT - Walk Through Alternative Node Configurations . . . 133
9.2.1.4 tCONST - Alteration of Constraints . . . 133
9.2.1.5 tSEP - Alteration of Separator Node . . . 134
9.2.1.6 tWALK - Walk Through a Data Model . . . 135
9.2.1.7 tWALKcsp - Walk Through the Constraint of a Data Model . . . 136
9.2.2 Stateless Disruptors . . . 136
9.2.2.1 ADD - Add Data Within a Node . . . 136
9.2.2.2 OP - Perform Operations on Nodes . . . 137
9.2.2.3 MOD - Modify Node Contents . . . 138
9.2.2.4 CALL - Call Function . . . 138
9.2.2.5 NEXT - Next Node Content . . . 139
9.2.2.6 FIX - Fix Data Constraints . . . 139
9.2.2.7 ALT - Alternative Node Configuration . . . 140
9.2.2.8 C - Node Corruption . . . 140
9.2.2.9 Cp - Corruption at Specific Position . . . 141
9.2.2.10 EXT - Make Use of an External Program . . . 141
9.2.2.11 SIZE - Truncate . . . 141
9.2.2.12 STRUCT - Shake Up Data Structure . . . 142
9.2.2.13 COPY - Shallow Copy Data . . . 142

10 Generic Targets **143**
10.1 NetworkTarget . . . 143
10.2 LocalTarget . . . 145
10.3 SSHTarget . . . 145
10.4 PrinterTarget . . . 146

10.5 SIMTarget . . . 146
10.6 TestTarget . . . 146

11 Generic Probes and Backend **149**
11.1 Generic Backend . . . 150
11.1.1 SSH_Backend . . . 150
11.1.2 Serial_Backend . . . 150
11.1.3 Shell_Backend . . . 150
11.2 Generic Probes . . . 150
11.2.1 ProbePID . . . 150
11.2.2 ProbeMem . . . 150
11.2.3 ProbeCmd . . . 151

12 Useful Examples **153**
12.1 ZIP archive modification . . . 153

13 Fuddly API **155**
13.1 API Index . . . 155
13.2 framework package . . . 155
13.2.1 framework.basic_primitives module . . . 155
13.2.2 framework.data module . . . 155
13.2.3 framework.data_model module . . . 160
13.2.4 framework.node module . . . 163
13.2.5 framework.node_builder module . . . 190
13.2.6 framework.value_types module . . . 200
13.2.7 framework.generic_data_makers module . . . 219
13.2.8 framework.target_helpers module . . . 229
13.2.9 framework.targets.network module . . . 233
13.2.10 framework.targets.local module . . . 238
13.2.11 framework.targets.sim module . . . 239
13.2.12 framework.targets.ssh module . . . 239
13.2.13 framework.targets.printer module . . . 241
13.2.14 framework.targets.debug module . . . 242
13.2.15 framework.project module . . . 244
13.2.16 framework.operator_helpers module . . . 245
13.2.17 framework.logger module . . . 247
13.2.18 framework.monitor module . . . 249
13.2.19 framework.comm_backends module . . . 257
13.2.20 framework.tactics_helpers module . . . 261
13.2.21 framework.fuzzing_primitives module . . . 266
13.2.22 framework.encoders module . . . 271
13.2.23 framework.database module . . . 273
13.2.24 framework.scenario module . . . 276
13.2.25 framework.dmhelpers.generic module . . . 283
13.2.26 framework.dmhelpers.xml module . . . 288
13.2.27 framework.evolutionary_helpers module . . . 289
13.2.28 framework.knowledge.feedback_collector module . . . 291
13.2.29 framework.knowledge.feedback_handler module . . . 292
13.2.30 framework.knowledge.information module . . . 294
13.2.31 framework.constraint_helpers module . . . 296
13.2.32 framework.plumbing module . . . 298
13.2.33 libs.utils module . . . 302

14 Indices and tables **305**

Python Module Index 307

Index 309

Contents:

CHAPTER

ONE

OVERVIEW

Among the variety of complementary approaches used in the security evaluation of a *target* (*e.g.*, software, an embedded equipment, *etc.*), fuzz testing—abbreviated *fuzzing*—is widely recognized as an effective means to help discovering security weaknesses in a target.

Fuzzing is a software testing approach, which consists in finding design or implementation flaws by stepping outside the expectations the target may have relative to its input data, while looking out for any unexpected behavior. This approach strives to confuse the target in a way to specifically avoid rejection by possible conformity tests—performed by the target—while still having a chance to trigger more subtle bugs. For such purpose, various ways are worth considering like using malformed data, playing around the protocol sequencing, and so on. Fuzzing is similar to what is termed *fault injection* in the field of *dependability*.

`fuddly` is a fuzzing and data manipulation framework whose main objectives are:

- To allow users to build data model that:
  - mix very accurate representations for certain aspects with much coarser ones for others that are outside the focus of the testing; leaving open the way of refining the other parts should the need arise;
  - may be combined with each other;
  - enable to dissect raw data for analyzing them and enable to absorb them within the data model for manipulation;
  - enable to mix up generation and mutation fuzzing techniques.
- To represent the data in a way that simplify the process of fuzzing and especially to enable the implementation of elaborated transformations. By ''elaborated" we mean the capability to act on any data part (that is not necessarily contiguous) while preserving consistency of dependent parts if so desired. This amounts to allowing transformations to be articulated around syntactic criteria—*e.g.*, modification of an integer depending on the size of the field hosting it—or semantic ones—*e.g.*, alteration of a value regarding its meaning for a given data format or protocol, alteration of specific data sub-parts forming a sound group for a given data format or protocol.
- To automate the fuzzing process relying on various fuddly's sub-systems enabling: the communication with the target, to follow and monitor its behavior and to act accordingly (*e.g.*, deviate from the protocol requirements like sequencing, timing constraints, and so on), thanks to data model search and modification primitives, while recording every piece of information generated during this process and enabling to replay it.

CHAPTER

TWO

TUTORIAL

In this tutorial we will begin with the basic UI of `fuddly`. Then we will see how to use `fuddly` directly from an advanced python interpreter like `ipython`. Finally, we will walk through basic steps to create a new data model and the way to define specific disruptors.

2.1 Using `fuddly` simple UI: `Fuddly Shell`

A simple UI—called Fuddly Shell—allows to interact with `fuddly` in an easy way. In this tutorial we present the usual commands that can be used during a fuzzing session. But first we have to launch it by running the `<root of fuddly>/fuddly_shell.py` script.

Note: This script basically does the following:

```
fmk = FmkPlumbing()
fmk.start()

shell = FmkShell("Fuddly Shell", fmk)
shell.cmdloop()
```

2.1.1 Start a Fuzzing Session

After running this script you should be prompted with something like this:

```
============================================================[ Data Models ]==
>>> Look for Data Models within 'data_models' directory
*** Found Data Model: 'mydf' ***
*** Found Data Model: 'example' ***
>>> Look for Data Models within 'data_models/protocols' directory
*** Found Data Model: 'usb' ***
>>> Look for Data Models within 'data_models/file_formats' directory
*** Found Data Model: 'zip' ***
*** Found Data Model: 'png' ***
*** Found Data Model: 'pdf' ***
*** Found Data Model: 'jpg' ***
============================================================[ Projects ]==
>>> Look for Projects within 'projects/specific' Directory
*** Found Project: 'usb' ***
```

(continues on next page)

(continued from previous page)

```
>>> Look for Projects within 'projects/generic' Directory
*** Found Project: 'standard' ***
=============================================[ Fuddly Home Information ]==

 --> data folder: ~/.local/share/fuddly/
 --> contains: - fmkDB.db, logs, imported/exported data, ...
               - user projects and user data models, ...
 --> config folder: ~/.config/fuddly/

-=[ Fuddly Shell ]=- (with Fuddly FmK 0.30)

>>
```

Note that `fuddly` looks for *Data Model* files (within `data_models/`) and *Project* files (within `projects/`) during its initialization. A *Project* file is used to describe the targets that can be tested, the logger behaviour, and optionally specific monitoring means as well as some scenarios and/or virtual operators.

Note: Projects and data models files are retrieved either from `<root of fuddly>/{projects,data_models}/` or from `<fuddly data folder>/{projects,data_models}/`.

Note that when the Fuddly shell is launched, the path of the fuddly data folder is displayed as well as its configuration folder.

See also:

To create a new project file, and to describe the associated components refer to *Defining a Project Environment*.

Once loaded, a project can be used with any data models. Basically, that means you can send any kind of data (among the defined ones) to any target described within your project file.

Let's start by loading the `standard` project which define some targets to play with:

```
>> load_project standard
```

You can look at the defined targets by issuing the following command:

```
>> show_targets

-=[ Available Targets ]=-

[0] EmptyTarget [ID: 307144]
[1] LocalTarget [Program: display]
[2] LocalTarget [Program: okular]
[3] LocalTarget [Program: unzip, Args: -d ~/.local/share/fuddly/workspace/]
[4] PrinterTarget [IP: 127.0.0.1, Name: PDF]
[5] NetworkTarget [localhost:54321, localhost:12345]
```

By default, the `EmptyTarget` is selected in order to let you experiment without a real target. But let's say you want to fuzz the `unzip` program. You first have to select it:

```
>> load_targets 3
```

Note: You can also load several targets at the same time if you want to sequence different actions through various

systems or on the same system but through different kinds of interfaces (represented by different targets). To do it, provide a list of target IDs to the `load_targets` command. For instance to load the targets 1, 4 and 5, issue the command:

```
>> load_targets 1 4 5
```

See also:

In order to define new targets, look at *Defining the Targets*.

See also:

`Target` (*framework.target_helpers.Target*) configuration cannot be changed dynamically within `Fuddly Shell`. But you can do it through any python interpreter, by directly manipulating the related `Target` object. Look at *Using fuddly Through Advanced Python Interpreter*.

You also need to choose a *Data Model* that you want to use with the selected target. For that purpose you can first list the available data models:

```
>> show_data_models

-=[ Data Models ]=-

[0] mydf
[1] usb
[2] zip
[3] png
[4] pdf
[5] jpg
 ...
```

As we select the `unzip` program as a target, we may want to perform ZIP fuzzing ;) Thus we select this data model by issuing the following command:

```
>> load_data_model zip
```

And then we launch the loaded project and all the components by issuing the following command:

```
>> launch

*** Data Model 'zip' loaded ***
*** Logger is started ***
*** Target initialization: (0) EmptyTarget [ID: 307144] ***
*** Monitor is started ***

*** [ Sending delay = 0.00s ] ***
*** [ Number of data sent in burst = 1 ] ***
*** [ Target EmptyTarget [ID: 241984] health-check timeout = 10.0s ] ***
>>
```

Note: Note that just after the project is launched, some internal parameters are displayed, namely:

- The sending delay, which allows you to set a minimum delay between two data emission. (Can be changed through the command `set_delay`).

- The maximum number of data that will be sent in burst, thus ignoring the sending delay. (Can be changed through the command `set_burst`)
- The timeout value for checking target's health. (Can be changed through the command `set_health_check_timeout`)

Finally, you may prefer to directly launch your project thanks to the command `run_project`. Indeed, by using it, you will automatically trigger the commands we just talked about. Regarding the loaded data models it will initially load what is defined as default in the project file. In the case of the `standard` project, if you issue the following command:

```
>> run_project standard
```

the imaginary data model used by our tutorial (`mydf`) will be loaded and the default target will be chosen, namely the `EmptyTarget` (usefull for testing purpose) with the ID 0.

In order to run the project with the `unzip` target (ID 4), you will have to issue the following command:

```
>> run_project standard 4
```

Note: If you want to load other targets while your project is currently running, you should use the `reload_all` command (refer to *Reloading Data Models / Targets / . . .*)

Note: If you want to load another data model at any time while your project is launched, use simply the command `load_data_model` with the name of the data model you want to use, and that's all.

You can also load multiple data models through the command `load_multiple_data_model <dm_name_1> <dm_name_2> ... [dm_name_n]`, if you want to interact with a target with different data models simultaneously.

Note: The `help` command shows you every defined command within `Fuddly Shell`. You can also look at a brief command description and syntax by typing `help <command_name>`

2.1.2 Send Malformed ZIP Files to the Target (Manually)

2.1.2.1 How to Send a ZIP File

In order to send a ZIP file to a loaded target, type the following:

```
>> send ZIP [target ID]
```

In our case we previously only loaded the target ID 3 (linked to the `unzip` program). It means that issuing the following command with 3 as <target ID> will invoke the `unzip` program with a ZIP file:

```
__ setup generator 'g_zip' __

========[ 1 ]==[ 18/08/2015 - 19:24:34 ]=======================
### Target ack received at: None
### Step 1:
 |- generator type: ZIP | generator name: g_zip | User input: G=[ ], S=[ ]
### Data size: 47360 bytes
```

(continues on next page)

(continued from previous page)

```
### Emitted data is stored in the file:
/home/test/Tools/fuddly/exported_data/zip/2015_08_18_192434_00.zip
### Target Feedback:
...
>>
```

Note: If you don't provide a target ID on the command line, the one that will be used will be the first loaded one. Thus in our case, we can forget to specify the target ID.

Note: You can also send data to multiple targets at once (assuming that you enabled them at first), by providing the list of target IDs like the following command:

```
>> send ZIP 3 5
```

Note that a *framework.data_model.DataModel* can define any number of data types—to model for instance the various atoms within a data format, or to represent some specific use cases, ...

When a data model is loaded, a dynamic *Generator* is built for each data types registered within this data model. A *Generator* is the basic block for generating data. In our case, let us consult the *Generators* available for the ZIP data model:

```
>> show_generators

-=[ SPECIFIC GENERATORS ]=-

*** Available generators of type 'ZIP' ***
  name: g_zip (weight: 1, valid: True)
  generic args:
    |_ random
    |      | desc: make the data model random
    |      | default: False [type: bool]
    |_ determinist
    |      | desc: make the data model determinist
    |      | default: False [type: bool]
    |_ finite
    |      | desc: make the data model finite
    |      | default: False [type: bool]

*** Available generators of type 'ZIP_00' ***
  name: g_zip_00 (weight: 1, valid: True)
  generic args:
    |_ random
    |      | desc: make the data model random
    |      | default: False [type: bool]
    |_ determinist
    |      | desc: make the data model determinist
    |      | default: False [type: bool]
    |_ finite
    |      | desc: make the data model finite
```

(continues on next page)

(continued from previous page)

```
    |      | default: False [type: bool]

...
```

You can see that two generators are available for this data model. In this case—the ZIP data model—the first one will generate modeled ZIP archive based uniquely on the data model, whereas the other ones (`ZIP_00`, `ZIP_01`, ...) generate modeled ZIP archives based on the sample files available within the directory `<fuddly data folder>/imported_data/zip/`.

For each one of these generators, some parameters are associated:

- `random`: Enforce the generator to generate data in a random way;
- `determinist`: Enforce the generator to generate data in a deterministic way;
- `finite`: Enforce the generator to generate a finite number of data.

To send in a loop, five ZIP archives generated from the data model in a deterministic way—that is by walking through the data model—you can use the following command:

```
>> send_loop 5 ZIP(determinist=True) tWALK
```

We use for this example, the generic stateful disruptor `tWALK` whose purpose is to simply walk through the data model. Note that disruptors are chainable, each one consuming what comes from the left.

See also:

Refer to *How to Perform Automatic Modification on Data* for details on data makers chains.

Note that if you want to send data indefinitely until the generator exhausts (in our case `ZIP`) or a stateful disruptor (in our case `tWALK`) of the chain exhausts you should use `-1` as the number of iteration. In our case it means issuing the following command:

```
>> send_loop -1 ZIP(determinist=True) tWALK
```

And if you want to stop the execution before the normal termination (which could never happen if the `finite` parameter has not been set), then you have to issue a `SIGINT` signal to `fuddly` via `Ctrl-C` for instance.

2.1.2.2 How to Perform Automatic Modification on Data

In order to perform modification on a generated data, you can use *disruptors* (look at *Generic Disruptors*), which are the basic blocks for this task. You can look at the available disruptors—either specific to the data model or generic–by typing the command `show_disruptors`, which will print a brief description of each disruptor along with their parameters.

Note: The following command allows to briefly look at all the defined generators and disruptors (called data makers), usable within the frame of the current data model.

```
>> show_dmaker_types

===[ Generator Types ]=============================================================

 [ Specific ]
   | 4TG1, 4TG2, ABSTEST, ABSTEST2, ENC
   | EXIST_COND, LEN_GEN, MISC_GEN, OFF_GEN, SEPARATOR
```

(continues on next page)

(continued from previous page)

```
   | SHAPE, TESTNODE, ZIP, ZIP_00

===[ Disruptor Types ]============================================================

 [ Generic ]
   | ALT, C, COPY, Cp, EXT
   | FIX, MOD, NEXT, SIZE, STRUCT
   | tALT, tSEP, tSTRUCT, tTERM, tTYPE
   | tWALK
```

You can also chain disruptors in order to perform advanced transformations—kind of dataflow programming. You can mix generic/specific stateless/stateful disruptors, fuddly will take care of sequencing everything correctly.

Let's illustrate this with the following example:

```
>> send ZIP_00 C(nb=2:path="ZIP_00/file_list/.*/file_name") tTYPE(max_steps=50:order=True) SIZE(sz=256)

__ setup generator 'g_zip_00' __
__ setup disruptor 'd_corrupt_node_bits' __
__ cleanup disruptor 'd_fuzz_typed_nodes' __
__ setup disruptor 'd_fuzz_typed_nodes' __
__ setup disruptor 'd_max_size' __

========[ 1 ]==[ 20/08/2015 - 15:20:06 ]=======================
### Target ack received at: None
### Step 1:
 |- generator type: ZIP_00 | generator name: g_zip_00 | User input: G=[ ], S=[ ]
### Step 2:
 |- disruptor type: C | disruptor name: d_corrupt_node_bits | User input: G=[ ], S=[nb=2,path='ZIP_00/file_list/.*/file_name']
 |- data info:
    |_ current fuzzed node: ZIP_00/file_list/file:3/header/file_name/cts
    |_ orig data: b'photo-photo-paysage-norvege.png'
    |_ corrupted data: b'photo-\xf8hoto-paysage-norvege.png'
    |_ current fuzzed node: ZIP_00/file_list/file:2/header/file_name/cts
    |_ orig data: b'hello.pdf'
    |_ corrupted data: b'hello.pd\xf6'
### Step 3:
 |- disruptor type: tTYPE | disruptor name: d_fuzz_typed_nodes | User input: G=[max_steps=50], S=[order=True]
 |- data info:
    |_ model walking index: 1
    |_  |_ run: 1 / -1 (max)
    |_ current fuzzed node:     ZIP_00/file_list/file/header/common_attrs/version_needed
    |_  |_ value type:         <framework.value_types.Fuzzy_INT16 object at␣
↪0x7fbf961e5250>
    |_  |_ original node value: b'1400' (ascii: b'\x14\x00')
    |_  |_ corrupt node value:  b'1300' (ascii: b'\x13\x00')
### Step 4:
 |- disruptor type: SIZE | disruptor name: d_max_size | User input: G=[ ], S=[sz=256]
 |- data info:
```

(continues on next page)

(continued from previous page)

```
34      |_ orig node length: 1054002
35      |_ right truncation
36      |_ new node length: 256
37  ### Data size: 256 bytes
38  ### Emitted data is stored in the file:
39  /home/test/Tools/fuddly/exported_data/zip/2015_08_20_152006_00.zip
40  >>
```

After the command is issued, fuddly will ask the generator `ZIP_00` to generate a modeled ZIP archive and then will provide the outcomes to the following disruptor `C`. At this moment, fuddly will disable temporarily the generator, as the generated data need to be fully consumed first.

The disruptor `C` will then be executed to consume the generated data. This disruptor performs basic corruption within the modeled data (it randomly chooses nodes of the graph-based modeled data and perform random bit corruption on them). You can see that some parameters are also given to it, namely: `nb` and `path`. These parameters are specific to this disruptor. The first one asks it to choose only two nodes and the second one restrict the set of nodes thanks to a regular expression that selects the root paths from which the terminal nodes to corrupt can be chosen.

Note: As the data model of `fuddly` is built on directed graphs, we call *paths* in `fuddly` the graph paths of the graph representing the data. For more information on fuddly data model refer to *Data Modeling*.

In order to select nodes in the graph from the root or another node, different criteria (syntactic & semantic) can be provided to `fuddly`'s low-level primitives. One of this criteria is *paths*, and the syntax defined to represent paths is similar to the one of filesystem paths. Each path are represented by a python string, where node identifiers are separated by /'s. For instance: `'ZIP/file_list/file:2/header'`, is a path from the root of a modeled ZIP archive to the *header* of its second file.

In this case we even restricted the nodes to be only the `file_name` nodes among all the files of the ZIP archive, as you can see on lines 16 & 19.

See also:

If you want to see an ASCII representation of the data, in order to grasp the way the graph is built, issue the command `show_data` after the generation process. It will depict something like what is shown *here under*.

Note: Parameters are given to data makers (generators/disruptors) through a tuple wrapped with the characters `(` and `)` and separated with the character `:`. Syntax:

```
data_maker_type(param1=val1:param2=val2)
```

After `C` has performed its corruption, fuddly gets the result and provides it to `tTYPE`. This disruptor is stateful, so it could outputs many different data from the one provided to it. In this specific case, it will walk the graph representing the data and generate new samples each time it encounter a typed terminal node. In the *previous run*, we see on line 30 that the original value of the terminal node `../version_needed` (a little-endian UINT16) has been altered to `1300` from the original value `1400`—which are the hexadecimal encoded representation of the integer. Basically, the disruptor performed a decrement by one of this integer. On the *next run*—line 16—you can see that this disruptor performs an increment by one instead of. And it will change this integer until he has no more cases—these cases are based on the syntactic & semantic properties provided within the ZIP data model. Afterwards, it will go on with the next node.

Note: Stateless disruptors output exactly one data for each data provided as input.


[ZIP_00]
----
[0] ZIP_00 [NonTerm]
__(1) ZIP_00/start_padding [String] size=0B
| _ raw: ''
__[1] ZIP_00/file_list [NonTerm]
| __[2] ZIP_00/file_list/file [NonTerm]
| | __[3] ZIP_00/file_list/file/header [NonTerm]
| | | __(4) ZIP_00/file_list/file/header/sig [UINT32_le] size=4B
| | | | _ 67324752 (0x4034B50)
| | | | _ raw: 'PK\x03\x04'
| | | __[4] ZIP_00/file_list/file/header/common_attrs [NonTerm]
| | | | __(5) ZIP_00/file_list/file/header/common_attrs/version_needed [UINT16_le] size=2B
| | | | | _ 20 (0x14)
| | | | | _ raw: '\x14\x00'
| | | | __(5) ZIP_00/file_list/file/header/common_attrs/gp_bit_flag [UINT16_le] size=2B
| | | | | _ 2 (0x2)
| | | | | _ raw: '\x02\x00'
| | | | __(5) ZIP_00/file_list/file/header/common_attrs/compression_method [UINT16_le] size=2B
| | | | | _ 8 (0x8)
| | | | | _ raw: '\x08\x00'
| | | | __(5) ZIP_00/file_list/file/header/common_attrs/last_mod_time [UINT16_le] size=2B
| | | | | _ 42684 (0xA6BC)
| | | | | _ raw: '\xbc\xa6'
| | | | __(5) ZIP_00/file_list/file/header/common_attrs/last_mod_date [UINT16_le] size=2B
| | | | | _ 17799 (0x4587)
| | | | | _ raw: '\x87E'
| | | | __(5) ZIP_00/file_list/file/header/common_attrs/crc32 [UINT32_le] size=4B
| | | | | _ 2483681496 (0x9409F8D8)
| | | | | _ raw: '\xd8\xf8\t\x94'
| | | | __(5) ZIP_00/file_list/file/header/common_attrs/compressed_size [UINT32_le] size=4B
| | | | | _ 98881 (0x18241)
| | | | | _ raw: 'A\x82\x01\x00'
| | | | __(5) ZIP_00/file_list/file/header/common_attrs/uncompressed_size [UINT32_le] size=4B
| | | | | _ 99439 (0x1846F)
| | | | | _ raw: 'o\x84\x01\x00'
| | | | __(4) ZIP_00/file_list/file/header/file_name_length [UINT16_le] size=2B
| | | | | _ 30 (0x1E)
| | | | | _ raw: '\x1e\x00'
| | | | __(4) ZIP_00/file_list/file/header/extra_field_length [UINT16_le] size=2B
| | | | | _ 0 (0x0)
| | | | | _ raw: '\x00\x00'
| | | | __[4] ZIP_00/file_list/file/header/file_name [GenFunc | node_args: ZIP_00/file_list/file/header/file_name_length]
| | | | | __(5) ZIP_00/file_list/file/header/file_name/cts [String] size=30B
| | | | | _ raw: 'Fond-ecran-paysage-gratuit.jpg'
| | | | __[4] ZIP_00/file_list/file/header/extra_field [GenFunc | node_args: ZIP_00/file_list/file/header/extra_field_length]
| | | | __(5) ZIP_00/file_list/file/header/extra_field/cts [String] size=0B
| | | | _ raw: ''
| | | __[3] ZIP_00/file_list/file/data [GenFunc | node_args: ZIP_00/file_list/file/header/common_attrs/compressed_size]
| | | __(4) ZIP_00/file_list/file/data/cts [String] size=98881B
| | | _ raw: '\x9c\xbbwXSY\xf8-\x1czG\xc1\x04\xe9\xd2!\xa1H 4\x01\x05!\x10\x02\x01\xd2(\x01\x1d\x9aH" \x01AEq@\x8a\x1a\xe9\x84\x16\x8a\x94\x04\x08M\xc1\x00A\xd4
\xa1\x85jT M\x94\xaa(\xe0(\xa0c\x9b\xb9q\xe6w\xbf\xef\x9f\xef\x8f\xef\xde\x93\'\xe7\xec\xbcg\xef\x93\xe7M\xb2\xde\xb5\xd6>;\xff\xcc\xfe\xf3\x1ap\x10\xe9\xe2\xe6\x02\x10\x10\x0
0\x00\x04\xf8\x0f\xc0?\xcb\x80 \xf8%b8\x00\xe0\xe6\x06\x01\x88\x03\x00\x00A\x00T@\x10 \xc6o\xe9\xf1\x9f\xc4\x1b\xc2\xbf\xba\x01L\x00\xffm\xae\xe7\xcf\x9f\x8d<\x03\x00\x08\x01\x
00\xb9b\x00a~\xc4\xd4\xcc\xccT\xb0\xea\xbf^\xc9\xfc\x80\xd0\xff\xb4\x83\x84\xfe\x1b\xf1\xcf\x02\xc0\t %..).&%)!)#-%#\xa7*/w\xe0\x80\x9c\x96\x922P\x15\xacc\x081\xd0\xd1\xd776s\x
b026\xb57\xd5\xd7\xb7\xf6\xb0\xb1w\x82#\xdc\x10\x86\x96^x0\x17\x9c\x83+\x02\xfe\xeb"\x02\xd222r\x07\xe44\xe5\xe55\xe1&\xfa&\xf0\xff\xe3\xed\x9f^\x80\x9c\xb8P\x90p\xb2\x90\x80\x
16@PW@HN\xe0\x9fA\xc0\x11\xfe\'!"\xf0\xef\xf679\x02\x04\x04\x85\x84ED\xc5\xc4%$\xa5\xf8\x1d:\x0e\x02\x04\x05\x84\x84\x04\x85\x85DD\x84\xf9\t\n$\xf1\xcf\x03\x84\xe5D\xe45\xa1\x8
e\xa2\x87\xd0\xc1bZd\xa0YJ^\xb5\xb8\xf6\xc9\xb6>\x10f\xe2O\x1d\xf3\x90\xd8\xeb\x12\x92\n\x87\x15\x95\x94u\xf5\xf4\r\xc0\x10\x98\x85\xa5\x95\xb5\xcd1\'g\xb8\x8b' ...
| | __[3] ZIP_00/file_list/file/data_desc [GenFunc | node_args: ZIP_00/file_list/file/header/common_attrs/gp_bit_flag, ZIP_00/file_list/file/header/common_attrs/crc32, ZI
P_00/file_list/file/header/common_attrs/compressed_size, ZIP_00/file_list/file/header/common_attrs/uncompressed_size]
| | __(4) ZIP_00/file_list/file/data_desc/no_data_desc [String] size=0B
| | _ raw: ''
| __[2] ZIP_00/file_list/file:2 [NonTerm]
| | __[3] ZIP_00/file_list/file:2/header [NonTerm]
| | | __(4) ZIP_00/file_list/file:2/header/sig [UINT32_le] size=4B
| | | | _ 67324752 (0x4034B50)
| | | | _ raw: 'PK\x03\x04'
| | | __[4] ZIP_00/file_list/file:2/header/common_attrs [NonTerm]

Stateful disruptors can output many data after being fed by only one data. When a stateful disruptor is called by `fuddly`—within a *chain* of disruptors—every data makers on its left are temporarily disabled. Thus, the next time the *chain* of disruptors is issued, the execution will begin directly with this stateful disruptor. And when this one has fully consumed its input, that is, when it cannot output any new data and handover to `fuddly`, the latter will re-enable the nearest left-side stateful disruptors that can provide new data, or the generator otherwise.

See also:

About *model walking* infrastructure of `fuddly` refer to *Defining Specific Disruptors*. Insights about how it deals with non-terminal changing nodes is provided.

About the parameters given to `tTYPE`, the generic one `max_steps=50` requests this disruptor to stop producing new data after a maximum of 50 for a unique input. The specific one `order=True` request it to strictly follow the data structure for producing its outcomes. Whether the order is set to `False` (or not given, as `False` is its default), the traversal to the data structure will be guided by other criteria depending on properties specified within the data model, especially the fuzz weight attribute that can be changed on any node and which defaults to 1. The bigger the value the higher the priority to be altered.

Note: To consult the help of a specific disruptor you can issue the command `show_disruptors <DISRUPTOR_TYPE>`

Finally, every data produced by `tTYPE` is given to the stateless disruptor `SIZE` whose purpose is to truncate the data if its size exceeds 256—as the parameter `sz` is equal to 256.

```
>> send ZIP_00 C(nb=2:path="$ZIP/file_list.*") tTYPE(max_steps=50:order=True)␣
↪SIZE(sz=256)

========[ 2 ]==[ 20/08/2015 - 15:20:08 ]=======================
### Target ack received at: None
### Initial Generator (currently disabled):
 |- generator type: ZIP_00 | generator name: g_zip_00 | User input: G=[ ], S=[ ]
  ...
### Step 1:
 |- disruptor type: tTYPE | disruptor name: d_fuzz_typed_nodes | User input: G=[max_
↪steps=50], S=[order=True]
 |- data info:
    |_ model walking index: 2
    |_  |_ run: 2 / -1 (max)
    |_ current fuzzed node:     ZIP_00/file_list/file/header/common_attrs/version_needed
    |_  |_ value type:         <framework.value_types.Fuzzy_INT16 object at␣
↪0x7fbf961e5250>
    |_  |_ original node value: b'1400' (ascii: b'\x14\x00')
    |_  |_ corrupt node value:  b'1500' (ascii: b'\x15\x00')
    |_ Data maker [#1] of type 'ZIP_00' (name: g_zip_00) has been disabled by this␣
↪disruptor taking over it.
    |_ Data maker [#2] of type 'C' (name: d_corrupt_node_bits) has been disabled by this␣
↪disruptor taking over it.
### Step 2:
 |- disruptor type: SIZE | disruptor name: d_max_size | User input: G=[ ], S=[sz=256]
 |- data info:
    |_ orig node length: 1054002
    |_ right truncation
    |_ new node length: 256
### Data size: 256 bytes
```

(continues on next page)

(continued from previous page)

```
### Emitted data is stored in the file:
/home/test/Tools/fuddly/exported_data/zip/2015_08_20_152008_00.zip
>>
```

On this *second command execution* you can see on lines 17-18 that the generator `ZIP_00` and the disruptor `C` have been disabled as explained before.

```
...
========[ 50 ]==[ 20/08/2015 - 15:20:11 ]======================
### Target ack received at: None
### Initial Generator (currently disabled):
 |- generator type: ZIP_00 | generator name: g_zip_00 | User input: G=[ ], S=[ ]
  ...
### Step 1:
 |- disruptor type: tTYPE | disruptor name: d_fuzz_typed_nodes | User input: G=[max_
↪steps=50], S=[order=True]
 |- data info:
    |_ model walking index: 50
    |_  |_ run: 6 / -1 (max)
    |_ current fuzzed node:     ZIP_00/file_list/file/header/common_attrs/uncompressed_
↪size
    |_  |_ value type:         <framework.value_types.Fuzzy_INT32 object at␣
↪0x7fbfec9e9048>
    |_  |_ original node value: b'6f840100' (ascii: b'o\x84\x01\x00')
    |_  |_ corrupt node value:  b'00000080' (ascii: b'\x00\x00\x00\x80')
    |_ Data maker [#1] of type 'ZIP_00' (name: g_zip_00) has been disabled by this␣
↪disruptor taking over it.
    |_ Data maker [#2] of type 'C' (name: d_corrupt_node_bits) has been disabled by this␣
↪disruptor taking over it.
### Step 2:
 |- disruptor type: SIZE | disruptor name: d_max_size | User input: G=[ ], S=[sz=256]
 |- data info:
    |_ orig node length: 1054002
    |_ right truncation
    |_ new node length: 256
### Data size: 256 bytes
### Emitted data is stored in the file:
/home/test/Tools/fuddly/exported_data/zip/2015_08_20_152011_15.zip
__ disruptor handover 'd_fuzz_typed_nodes' __
--------------------
| ERROR / WARNING |
--------------------
    (_ FMK [#DataUnusable]: The data maker (tTYPE) has returned unusable data. _)
    (_ FMK [#HandOver]: Disruptor 'd_fuzz_typed_nodes' (tTYPE) has yielded! _)
>>
```

If you go on issuing the same command, you will arrive at a point where `tTYPE` stops producing new data as seen *above* on lines 31 & 32. Thus, if you go on, this time the generator will be re-enabled to produce new data as well as the disruptor `C`, as seen *below* on line 3 & 5.

```
========[ 51 ]==[ 20/08/2015 - 15:26:19 ]======================
### Target ack received at: None
```

(continues on next page)

(continued from previous page)

```
### Step 1:
 |- generator type: ZIP_00 | generator name: g_zip_00 | User input: G=[ ], S=[ ]
### Step 2:
 |- disruptor type: C | disruptor name: d_corrupt_node_bits | User input: G=[ ], S=[nb=2,
↪path='ZIP_00/file_list/.*/file_name']
 |- data info:
    |_ current fuzzed node: ZIP_00/file_list/file:2/header/file_name/cts
    |_ orig data: b'hello.pdf'
    |_ corrupted data: b'hello\xafpdf'
    |_ current fuzzed node: ZIP_00/file_list/file/header/file_name/cts
    |_ orig data: b'Fond-ecran-paysage-gratuit.jpg'
    |_ corrupted data: b'Fond-ecran-paysage\xafgratuit.jpg'
### Step 3:
 |- disruptor type: tTYPE | disruptor name: d_fuzz_typed_nodes | User input: G=[max_
↪steps=50], S=[order=True]
 |- data info:
    |_ model walking index: 1
    |_  |_ run: 1 / -1 (max)
    |_ current fuzzed node:     ZIP_00/file_list/file/header/common_attrs/version_needed
    |_  |_ value type:         <framework.value_types.Fuzzy_INT16 object at␣
↪0x7fbfec9895f8>
    |_  |_ original node value: b'1400' (ascii: b'\x14\x00')
    |_  |_ corrupt node value:  b'1300' (ascii: b'\x13\x00')
### Step 4:
 |- disruptor type: SIZE | disruptor name: d_max_size | User input: G=[ ], S=[sz=256]
 |- data info:
    |_ orig node length: 1054002
    |_ right truncation
    |_ new node length: 256
### Data size: 256 bytes
### Emitted data is stored in the file:
/home/test/Tools/fuddly/exported_data/zip/2015_08_20_152619_00.zip
```

Last, to avoid re-issuing the same command for each time you want to send a new data, you can use the `send_loop` command as follows:

```
>> send_loop <NB> ZIP_00 C(nb=2:path="ZIP_00/file_list/.*") tTYPE(max_
↪steps=50:order=True) SIZE(sz=256)
```

where <NB> shall be replaced by the maximum number of iteration you want before fuddly return to the prompt. Note that it is a maximum; in our case it will stop at the 50 th run because of `tTYPE`. Note that you can also use the special value -1 to loop indefinitely or until a data maker is exhausted. In such situation, if you want to interrupt the looping, just use `Ctrl+C`.

2.1.2.3 Resetting & Cloning Disruptors

Whether you want to use generators or disruptors that you previously used in a *data maker chain*, you would certainly need to reset it or to clone it. Indeed, every data maker has an internal sequencing state, that remember if it has been disabled (and for generators it may also keeps the *seeds*). Thus, if you want to reuse it, one way is to reset it by issuing the following command:

```
>> reset_dmaker <dmaker_type>
```

where <dmaker_type> is the data maker to reset, for instance: ZIP_00, tTYPE, ...

You can also reset all the data makers at once by issuing the following command:

```
>> reset_all_dmakers
```

Note: In the case where the original data (i.e., the pristine generated data that does not get changed by any disruptor) is asked to be preserved (for instance by using the command send_loop_keepseed), for repeatability purpose (when issuing the same command again), using the previous command will also remove this original data. Thus you could prefer to use the command cleanup_dmaker that will only reset the sequencing state, without resetting the seed (i.e., the original data).

Note that keeping such *seeds* may consume a lot of memory at some point. Moreover, they may only be useful for non-determinist data model.

Another way that can reveal itself to be useful (especially within framework.tactics_helper.Operator— refer to *Defining Operators*) is to clone a data maker. By doing so, you have a new independent data maker that can be used in another *data maker chain*. To create a clone, just add #ID (where ID shall be replaced by a string up to 20 alphanumeric characters or underscore) to an existing data maker. For instance, issuing the following command, after having issuing the commands from the section *How to Perform Automatic Modification on Data*, won't alter anything regarding the state of the cloned data makers:

```
>> send ZIP_00#new tTYPE#new
```

2.1.2.4 Reloading Data Models / Targets / ...

If during a test session you want to perform a modification within the data model without restarting fuddly, you can simply edit the data model with your favorite editor, and after saving it, issue the command reload_data_model at the Fuddly Shell prompt.

If you also want to modify the target abstraction or operators or probes, ..., you have to reload every fuddly subsystems. To do so, you only need to issue the command reload_all.

Now, imagine that you want to switch to a new target already registered, simply issue the command reload_all [target_ID1 .. target_ID2], where target IDs are picked up through the IDs displayed by the command show_targets

Finally, if you want to switch to a new data model while a project is already launched, simply issue the command load_data_model <data_model_name> to let fuddly do the job for you.

2.1.3 Use an Operator to Send Malformed Data

`Operators` (`framework.tactics_helper.Operator`) are useful to automate the fuzzing process, that is to automatically collect target feedback when its worth it, to automatically save test cases that affect the target and to automatically decide on the following steps based on thoughtful criteria.

Let's take the example of a fuzzing operator defined in the `standard` project, and use it to fuzz JPG files and send them to the `display` program—target number 3.

See also:

To define your own operators refer to *Defining Operators*.

First, we need to launch the project `standard` and to specify the target number 3. You can do it in one line by issuing the following command:

```
>> run_project standard 3
```

The last parameter of is the identifier of the target. It's a shortcut to what have been presented in section *Start a Fuzzing Session*. If you issue the command `show_targets` you will notice the enabled target as it is highlighted in the console, like you can see in the figure *bellow*.

```
>> show_targets

-=[ Available Targets ]=-

[0] EmptyTarget
[1] LocalTarget [Program: display]
[2] LocalTarget [Program: okular]
[3] LocalTarget [Program: unzip, Args: -d /home/tuxico/Projets/fuddly/workspace/]
[4] PrinterTarget [IP: 127.0.0.1, Name: PDF]
[5] NetworkTarget [localhost:54321, localhost:12345]
```

You can now load the JPG data model:

```
>> load_data_model jpg
```

Then, you can look at the available operators and learn about their parameters by issuing the command:

```
>> show_operators
```

This command will display the *following*:

To launch the operator `Op1` and limit to 5 the number of test cases to run, issue the command:

```
>> launch_operator Op1(max_steps=5)
```

This will trigger the Operator that will execute the `display` program with the first generated JPG file. It will look at `stdout` and `stderr` for error messages, or look for any crashes, and if such a situation occurs, will save the related JPG file under `exported_data/jpg/` and log everything under `logs/`. It will also try to avoid saving JPG files that trigger errors whose type has already been seen. Once the operator is all done with this first test case, it can plan the next actions it needs `fuddly` to perform for it. In our case, it will go on with the next iteration of a disruptor chain, basically `JPG(finite=True) tTYPE`.

```
>> show_operators

-=[ Operators ]=-

Op1

  generic args:
    |_ init
    |       | desc: make the model walker ignore all the steps until the provided
    |       |       one
    |       | default: 1 [type: int]
    |_ max_steps
    |       | desc: number of test cases to run
    |       | default: 20 [type: int]
  specific args:
    |_ path
    |       | desc: path of the target application (for LocalTarget's only)
    |       | default: /usr/bin/display [type: str]
    |_ mode
    |       | desc: strategy mode (0 or 1)
    |       | default: 0 [type: int]
```

2.1.4 Replay Data From a Previous Session

If you want to replay some data previously sent, you can either use the *workspace* where each emitted data are registered (in memory only) during a `fuddly` session, or if you quit `fuddly` in-between you can reload data from the `fuddly` database `fmkDB.db` (SQLite3).

To resend the data you just sent, issue the following command:

```
>> replay_last
```

But if you want to resend any data from the *workspace* you first have to store it to the *Data Bank*. To save the data you just sent, issue the following command:

```
>> register_last
```

To save all the *workspace* in the *Data Bank*, issue the following command:

```
>> register_wkspace
```

Then, if you want to look at the *Data Bank*, issue the command:

```
>> show_db
```

You will then be able to resend any data from the *Data Bank* thanks to its entry number (that is displayed by the previous command). For instance, if you want to resend the data registered in the 5th entry of the *Data Bank*, issue the command:

```
>> replay_db 5
```

Finally, if you want to resend data from previous sessions, you can do it by looking at the `DATA` table of the `fmkDB.db`, looking for the IDs that match the data you want to resend and use the command `fmkdb_fetch_data`. Let's say you want to load the data from ID 32 to ID 105, you will issue the following command:

```
>> fmkdb_fetch_data 32 105
```

That command will store these data to the *Data Bank*. From then on, you could use `show_db` and `replay_db` as previously explained.

Note: You can use disruptors with a `replay_*` command. However if these disruptors are stateful, you should issue the command only once. Then, if you want to walk through the stateful disruptor, you only have to switch to a `send`-like command, and use as generator name the string `NOGEN`

For instance:

```
>> replay_last tTYPE
>> send_loop -1 NOGEN tTYPE
```

2.2 Using `fuddly` Through Advanced Python Interpreter

To use `fuddly` within any python interpreter like `ipython`, you will need to issue the following commands:

```
from framework.plumbing import *

fmk = FmkPlumbing()
fmk.start()
```

From now on you can use `fuddly` through the object `fmk`. Every commands defined by `Fuddly Shell` (refer to *Start a Fuzzing Session*) are backed by a method of the class *framework.plumbing.FmkPlumbing*.

Here under some basic commands to launch the project `tuto`, a virtual testing target and the `ZIP` data model:

```
# To show the available projects
fmk.show_projects()

# Contains the list of all the Project objects available
fmk.prj_list

# Load the ``tuto`` project by name
fmk.load_project(name='tuto')

# Reference to the currently launched project, in our case ``tuto``
fmk.prj

# Show available targets for this project
fmk.show_targets()

# Select the target with ID ``7``
fmk.load_targets(7)

# To show all the available data models
fmk.show_data_models()

# Contains the list of all the DataModel objects available
fmk.dm_list

# Load the ZIP data model by name
fmk.load_data_model(name='zip')

# Reference to the currently loaded data model, in our case the ZIP one
```

(continues on next page)

(continued from previous page)

```
fmk.dm

# Launch the project and all the related components
fmk.launch()
```

Note: The previous commands used to load a project, targets and data models can be factorized in one line thanks to the following command:

```
# To launch the ``tuto`` project with the targets ID ``7`` and ``8``
# and the ZIP data model in one line
fmk.run_project(name='tuto', tg_ids=[7,8], dm_name='zip')
```

You can also change the timeout value used to retrieved feedback from the targets, as well as tuning the way this value has to be considered (a maximum value or a strict time slice).

```
fmk.set_feedback_timeout(1, tg_id=7)
fmk.set_feedback_mode(Target.FBK_WAIT_UNTIL_RECV, tg_id=7)
fmk.set_feedback_timeout(2, tg_id=8)
fmk.set_feedback_mode(Target.FBK_WAIT_FULL_TIME, tg_id=8)
```

The effect of this commands is summarized in a specific screen that can be displayed by issuing the command `fmk.show_fmk_internals()`:

```
-=[ FMK Internals ]=-

  [ General Information ]
                  FmkDB enabled: True
              Workspace enabled: True
                  Sending delay: 0.0
   Number of data sent in burst: 1
 Target(s) health-check timeout: 4.0

  [ Target Specific Information - (7) TestTarget [ID: 792104] ]
               Feedback timeout: 1
                  Feedback mode: Wait until the target has sent something back to us

  [ Target Specific Information - (8) TestTarget [ID: 792160] ]
               Feedback timeout: 2
                  Feedback mode: Wait for the full time slot allocated for feedback␣
↪retrieval
```

Other commands allowing you to perform some user code changes either in the project file or the data models and take them into account without restarting fuddly:

```
# Reload all sub-systems and data model definitions and choose the target ``0``
fmk.reload_all(tg_num=0)

# Reload the data model definitions
fmk.reload_dm()
```

Then, when everything is loaded, the following commands is an example on how target interaction can be performed:

```
# Show a list of the registered data type within the data model
fmk.show_atom_identifiers()
# Or
list(fmk.dm.atom_identifiers())

# Get an instance of the modeled data ZIP_00 which is made from the
# absorption of an existing ZIP archive within <fuddly data folder>/imported_data/zip/
dt = fmk.dm.get_atom('ZIP_00')

# Display the raw contents of the first generated element of the data type `dt`
# Its the flatten version of calling .get_value() on it. Note that doing so will
# freeze the data type to the generated output, no matter how many times you call
# these method on it
dt.to_bytes()

# Pretty print the current value. (if the data type is not already frozen,
# it will call g.get_value() on it)
dt.show()

# Unfreeze the data type to get a new value and then display it
dt.unfreeze()
dt.show()

# Send the current data, log it and save it
fmk.send_data_and_log(Data(dt))

# Perform a tTYPE disruption on it, but give the 5th generated
# cases and enforce the disruptor to strictly follow the ZIP structure
# Finally truncate the output to 200 bytes
action_list = [('tTYPE', UI(init=5, order=True)), ('SIZE', UI(sz=200))]
altered_data = fmk.process_data(action_list, seed=Data(dt))

# Send this new data and look at the actions that perform tTYPE and
# SIZE through the console or the logs
fmk.send_data_and_log(altered_data)
```

The last command will display something like this (with some color if you have the `xtermcolor` python library):

```
====[ 3 ]==[ 27/06/2019 - 12:07:19 ]============================================
### Step 1:
 |- disruptor type: tTYPE | disruptor name: sd_fuzz_typed_nodes | User input: [init=5,
↪order=True]
 |- data info:
    |_ model walking index: 4
    |_  |_ run: 4 / -1 (max)
    |_ current fuzzed node:     ZIP_00/file_list/file/header/common_attrs/version_needed
    |_  |_ value type:          <framework.value_types.UINT16_le object at␣
↪0x7f2bcd49b160>
    |_  |_ original node value (hex): b'1403'
    |_  |                     (ascii): b'\x14\x03'
    |_  |_ corrupt node value  (hex): b'0000'
    |_                        (ascii): b'\x00\x00'
### Step 2:
```

(continues on next page)

(continued from previous page)

```
 |- disruptor type: SIZE | disruptor name: d_max_size | User input: [sz=200]
 |- data info:
    |_ orig node length: 595
    |_ right truncation
    |_ new node length: 200
### Data size: 200 bytes
### Emitted data is stored in the file:
<fuddly data folder>/exported_data/zip/2019_06_27_120719_00.zip
### FmkDB Data ID: 542

### Ack from 'TestTarget [ID: 725768]' received at: 2019-06-27 12:07:19.071751
### Feedback from 'TestTarget [ID: 725768]' (status=0):
CRC error
```

The previous commands can be factorized through the method *framework.plumbing.FmkPlumbing.process_data_and_send()*

For instance fuzzing the targets 7 and 8 simultaneously (that handle ZIP format) until exhaustion of test cases can be done thanks to the following lines:

```
# Hereunder the chosen fuzzing follow a 2-step approach:
# 1- the disruptor tTYPE is called on the seed and starts from the 5th test case
# 2- a trailer payload is added at the end of what is generated previsouly

dp = DataProcess([('tTYPE', UI(deep=True, init=5)),
                  ('ADD', UI(raw='This is added at the end'))],
                 seed='ZIP_00')

fmk.process_data_and_send(dp, max_loop=-1, tg_ids=[7,8])
```

We did not discuss all the methods available from `framework.plumbing.FmkPlumbing`but you should now be more familiar with :class:`framework.plumbing.FmkPlumbing` and go on with its exploration.

Finally, in order to exit the framework, the following method should be called (otherwise, various threads would block the correct termination of the framework):

```
fmk.stop()
```

For more information on how to manually make modification on data, refer to the section *Defining Specific Disruptors*

2.3 Implementing a Data Model and Defining a Project Environment

2.3.1 Data Modeling

2.3.1.1 Overview

Within fuddly data representation is performed through the description of a directed acyclic graph whose terminal nodes describe the different parts of a data format and the arcs—which can be of different kinds—capture its structure. This graph includes syntactic and semantic information of the data format. Using a graph as a data model enables to represent various kind of data format with flexibility. By flexibility we mean the possibility to mix accurate representations for certain aspects with much coarser ones for others—e.g., modeling accurately only the data parts which are assumed to be complex to handle by the target—and a high-level of expressiveness.

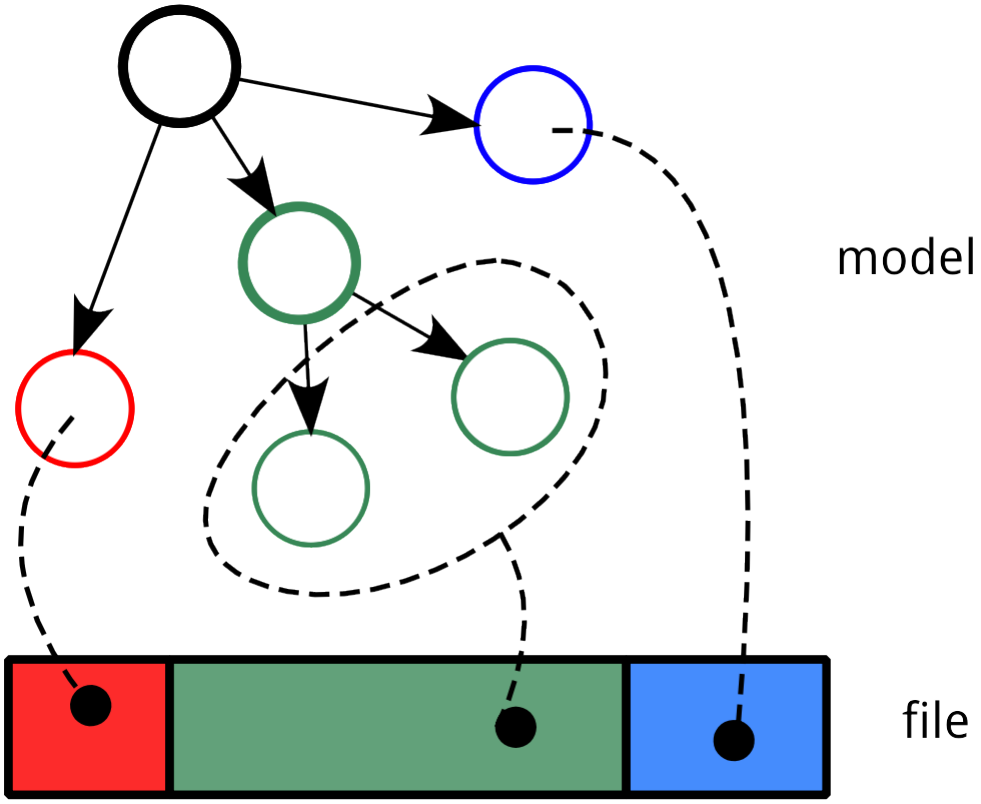


Fig. 1: Data Representation

From this model, data can be generated (look at the figure *Data Generation*) and existing raw data can be absorbed. This latter operation is a projection of the existing raw data within the data model (see the example *ZIP archive modification* and also the section *Absorption of Raw Data that Complies to the Data Model*). Data generation allows to create data that conforms to the model if we want to iteract correctly with the target, or to create degenerate data if we want to assess target robustness. Data absorption can allow to generate data from existing ones if the model is not accurate enough to generate correct data by itself; or to understand the target outputs in order to interact correctly with it or not.

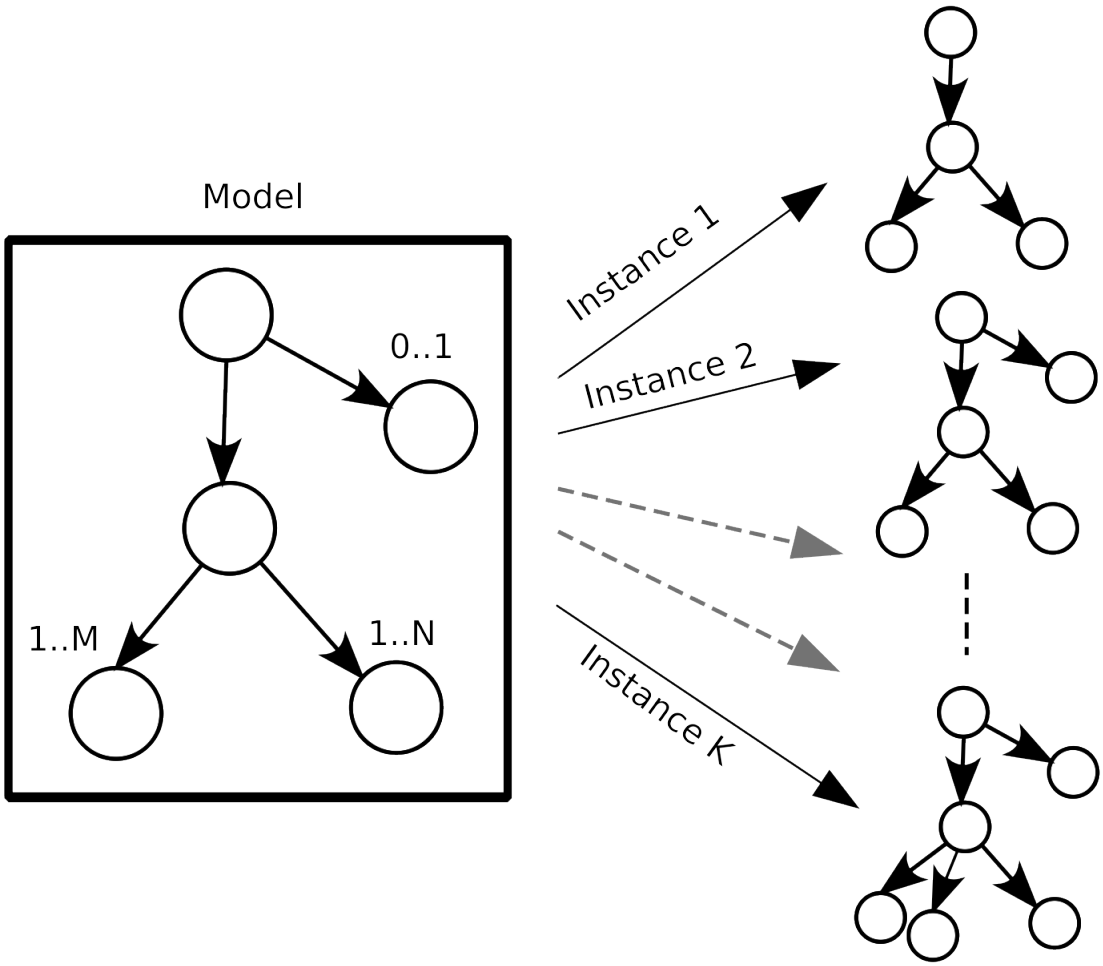


Fig. 2: Data Generation

Generating data boils down to walk the graph that model the data format. After each traversal, a data is produced and each traversal make the graph evolving, in a deterministic or random way depending on your intent. Graph walking is also a way to perform node alteration on the fly (through entities called *disruptors*).

See also:

Refer to *Defining Specific Disruptors* to learn how to perform modification of data generated from the model. Refer to *How to Perform Automatic Modification on Data* in order to play with existing generic disruptors within the frame of the `fuddly` shell.

Different kinds of node are defined within fuddly in order to model data:

- Terminal nodes with typed-value contents (e.g., `UINT16`, `BitField`, `String`, ...)
- Non-terminal nodes that are used to define the data format structure. They put in order the different parts of a data format, and can even specify a grammar to express a more complex assembly.

- *Generator* nodes that are used to dynamically generate a part of the graph according to other nodes (from within the graph itself or not) and/or other criteria provided as parameters.

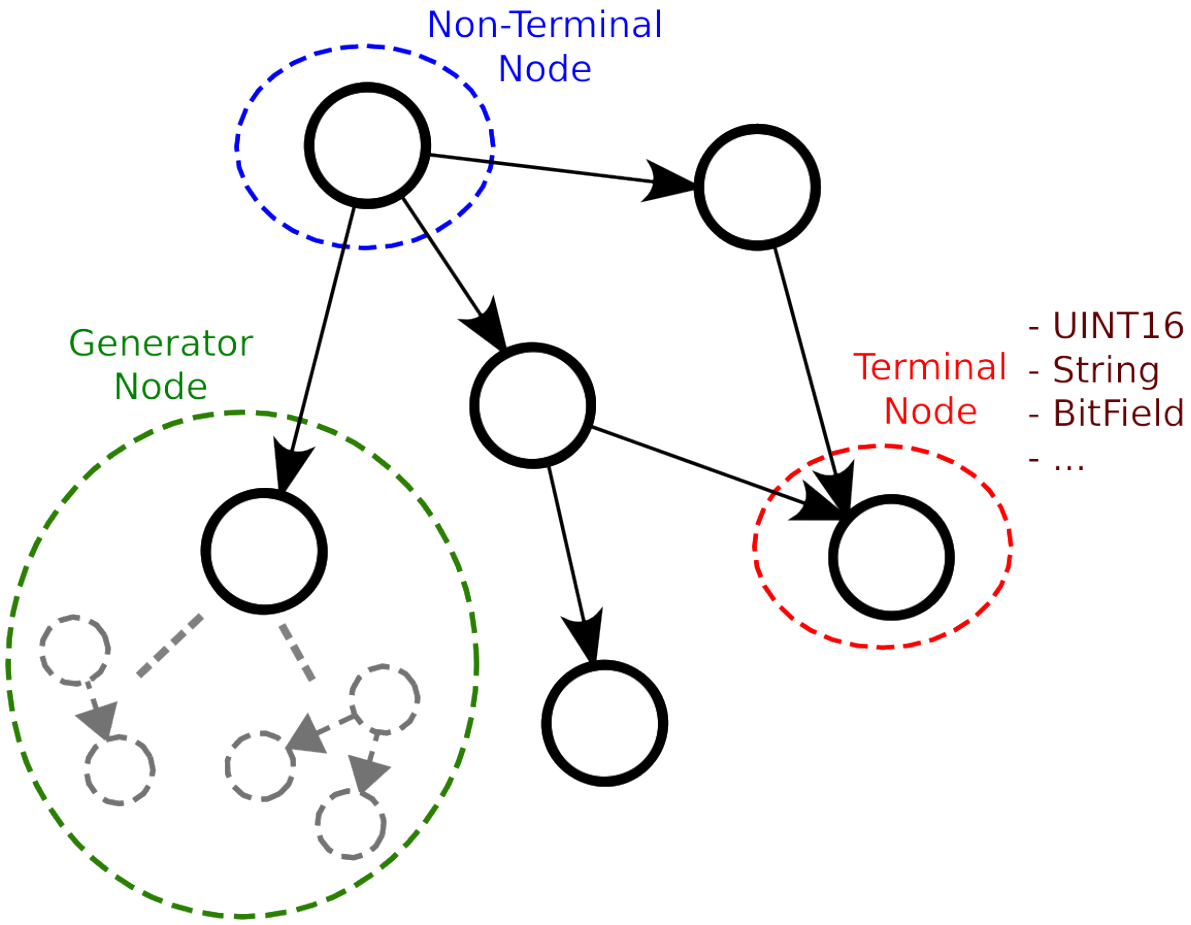


Fig. 3: Node Types

The structure of a data format is grasped by the links between the graph nodes. Within `fuddly` data model, we distinguish three kinds of links:

- Parent-child links which define a basic structure between the graph nodes. They are ruled by non-terminal nodes.
- Links associated to specific criteria that condition some part of the graph. For instance, node generation can be associated to the existence of another one; different node set can be synchronized relatively to their cardinality.
- Links defined between generator nodes and their parameter nodes. They are especially useful when a complex relationship exist between multiple nodes. The generator nodes are then used to rule this relationship by defining it through a function.

Additionally, for each node can be defined alternative configurations, enabling for instance to dynamically change a terminal node in a non-terminal node or a generator node. These configurations can be added dynamically and switched at any times even during the graph traversal. This feature can be leveraged to capture different facets of a data format within the same data model; while offering the possibility to work on only one view at a time. It can also be useful for absorption. Indeed, this operation can require to model some part of the data format in a way different from the one took on for the generation. The alternative configurations enable to aggregate these differences within the same data model.

Finally, it is also possible to associate various kind of attributes to the nodes:

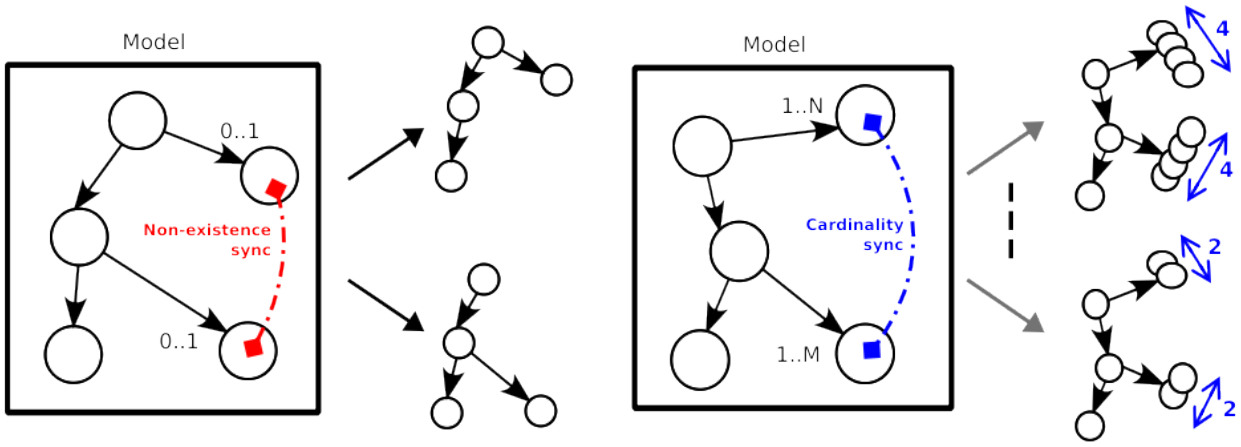


Fig. 4: Node Constraints

- classic ones like Mutable, Determinist, Finite, ...
- semantic ones that allows to group nodes based on some specific meanings (for instance a PDF page), in order to enable higher level data manipulation.
- user-defined ones for specifying specific semantics to the nodes to enable enhanced data modification.

2.3.1.2 A First Example

In order to create a data model, `fuddly`'s low-level primitives can be used, or more simply the high-level infrastructure that create the model from kind of JSON representation. For complex case, the two approaches could be complementary. Moreover data models can also use other data models whether the need arises.

Let's look at the following example which is a limited description of the PNG data format:

```
png_desc = \
{'name': 'PNG_model',
 'contents': [
    {'name': 'sig',
     'contents': String(values=[b'\x89PNG\r\n\x1a\n'], size=8)},
    {'name': 'chunks',
     'qty': (2,-1),
     'contents': [
          {'name': 'len',
           'contents': UINT32_be()},
          {'name': 'type',
           'contents': String(values=['IHDR', 'IEND', 'IDAT', 'PLTE'], size=4)},
          {'name': 'data_gen',
           'contents': lambda x: Node('data', value_type= \
                       String(size=x[0].get_raw_value())),
           'node_args': ['len']},
          {'name': 'crc32_gen',
           'contents': CRC(vt=UINT32_be, clear_attrs=[MH.Attr.Mutable]),
           'node_args': ['type', 'data_gen']}
     ]}
 ]}
```

In short, we see that the root node is `PNG_model`, which is the parent of the terminal node `sig` representing PNG file signature (lines 4-5) and the non-terminal node `chunks` representing the file's chunks (lines 6-20)[1]. This latter node describe the PNG file structure by defining the chunk contents in lines 9-19—in this very simplistic data model, chunk types are not distinguished, but it can easily be expanded—and the number of chunks allowed in a PNG file in line 7—from `2` to `-1` (meaning infinity).

See also:

For detailed information on how to describe a data format and getting the list of the usable keywords refer to *Data Model Patterns* and *Data Model Keywords*.

2.3.1.3 Defining the Imaginary MyDF Data Model

Assuming we want to model an imaginary data format called *MyDF*. Two files need to be created either within `<root of fuddly>/data_models/` or within `<fuddly data folder>/user_data_models/` (or within any subdirectory):

`mydf.py` Contain the implementation of the data model related to `MyDF` data format, **which is the topic of the current section**.

`mydf_strategy.py` Contain optional disruptors specific to the data model (*Defining Specific Disruptors*)

By default, `fuddly` will use the prefix `mydf` for referencing the data model. But it can be overloaded within the data model definition, as it is done in the following example (in line 8) which is a simple skeleton for `mydf.py`:

```
1  from framework.node import *
2  from framework.value_types import *
3  from framework.data_model import *
4
5  class MyDF_DataModel(DataModel):
6
7      file_extension = 'myd'
8      name = 'overload_default_name_if_you_wish'
9
10     def build_data_model(self):
11
12         # Data Type Definition
13         d1 = ...
14         d2 = ...
15         d3 = ...
16
17         self.register(d1, d2, d3)
18
19
20 data_model = MyDF_DataModel()
```

Note: All elements discussed during this tutorial, related to the data model `mydf`, are implemented within `tuto.py` and `tuto_strategy.py`. Don't hesitate to play with what are defined within, Either with `ipython` or `Fuddly Shell` (*Start a Fuzzing Session*).

In this skeleton, you can notice that you have to define a class that inherits from the *framework.data_model.DataModel* class, as seen in line 5. The definition of the data types of a data format will be written in python within the method *framework.data_model.DataModel.build_data_model()*. In the previous listing, the data types (also

[1] These chunks are information blocks that compose every PNG file.

called *atoms*) are represented by `d1`, `d2` and `d3`. Once defined, they should be registered within the data model, by calling *framework.data_model.DataModel.register()* on them.

Note: In the frame of your data model if you want to instantiate atoms from samples:

- Add your samples there: `<fuddly data folder>/imported_data/<NAME of DM>/`
- Within the method *framework.data_model.DataModel.build_data_model()*, and once you defined your atoms, call the method *framework.data_model.DataModel.register_atom_for_decoding()* to register the atom that will be used to model your samples. (To perform this action the framework leverages the node absorption mechanism – *Absorption of Raw Data that Complies to the Data Model*.) For a usage example, refer to the ZIP data model.
- Finally, the next time you load your data model, you will have your samples *absorbed* and available through specific Generators automatically created for you.

If you need more flexibility in this sample absorption process, you should overwrite the method *framework.data_model.DataModel._atom_absorption_additional_actions()* as illsutrated by the JPG data model.

Finally, if you need even more flexibility in order to create atoms from samples, because node absorption is not satisfactory in your context, then you could overload the method *framework.data_model.DataModel._create_atom_from_raw_data_specific()*. Refer to the JSON data model for an illustration, where this method is overloaded in order to create either atoms that represent JSON schemas or atoms that model some JSON data; depending on the JSON files provided in `<fuddly data folder>/imported_data/json`.

Note: The method *framework.data_model.DataModel.register_atom_for_decoding()* is also leveraged by the decoding feature of the class *framework.data_model.DataModel*, which is implemented by the method *framework.data_model.DataModel.decode()*.

Indeed, the decoding feature will look for a valid atom for performing the absorption of the provided binary string in order to be able to decode it. And this search depends on the atoms you registered.

The data model decoding feature can be used for different purposes. It is leveraged for instance by the `Fmkdb` toolkit (refer to *Data Analysis*).

For briefly demonstrating part of fuddly features to describe data formats, we take the following example whose only purpose is to mix various constructions, and value types.

See also:

For a more thorough description of the patterns that can be used to describe data formats, refer to *Data Model Patterns* and *Data Model Keywords*.

See also:

For a list and description of the currently defined value types refer to *Value Types*.

```
1  d1 = \
2  {'name': 'TestNode',
3   'contents': [
4
5        # block 1
6        {'section_type': MH.Ordered,
7         'duplicate_mode': MH.Copy,
8         'contents': [
9
```

(continues on next page)

(continued from previous page)

```
            {'contents': BitField(subfield_sizes=[21,2,1], endian=VT.BigEndian,
                                  subfield_values=[None, [0b10], [0,1]],
                                  subfield_val_extremums=[[500, 600], None, None]),
             'name': 'val1',
             'qty': (1, 5)},

            {'name': 'val2'},

            {'name': 'middle',
             'custo_set': MH.Custo.NTerm.FrozenCopy,
             'custo_clear': MH.Custo.NTerm.MutableClone,
             'contents': [{
                 'section_type': MH.Random,
                 'contents': [

                     {'contents': String(values=['OK', 'KO'], size=2),
                      'name': 'val2',
                      'qty': (1, -1)},

                     {'name': 'val21',
                      'clone': 'val1'},

                     {'name': 'USB_desc',
                      'import_from': 'usb',
                      'data_id': 'STR'},

                     {'contents': lambda x: Node('cts', values=[x[0].to_bytes() \
                                                                + x[1].to_bytes()]),
                      'name': 'val22',
                      'node_args': [('val21', 2), 'val3']}
                 ]}]},

            {'contents': String(max_sz = 10),
             'name': 'val3',
             'sync_qty_with': 'val1',
             'alt': [
                 {'conf': 'alt1',
                  'contents': SINT8(values=[1,4,8])},
                 {'conf': 'alt2',
                  'contents': UINT16_be(min=0xeeee, max=0xff56),
                  'determinist': True}]}
        ]},

        # block 2
        {'section_type': MH.Pick,
         'weights': (10,5),
         'contents': [
             {'contents': String(values=['PLIP', 'PLOP'], size=4),
              'name': 'val4'},

             {'contents': SINT16_be(values=[-1, -3, -5, 7]),
              'name': 'val5'}
```

(continues on next page)

(continued from previous page)

```
62          ]},
63
64          # block 3
65          {'section_type': MH.FullyRandom,
66           'contents': [
67               {'contents': String(values=['AAA', 'BBBB', 'CCCCC']),
68                'name': ('val21', 2)},
69
70               {'contents': UINT8(values=[2, 4, 6, 8]),
71                'qty': (2, 3),
72                'name': ('val22', 2)}
73           ]}
74    ]}
```

At first glance, the data model is composed of three parts: *block 1* (lines 6-50), *block 2* (lines 53-61) and *block 3* (lines 64-72). Within these blocks, various constructions are used. Below, some insights:

line 6, line 22, line 54, line 65 The keyword `section_type` allows to choose the order to be enforce by a non-terminal node to its children. `MH.Ordered` specifies that the children should be kept strictly in the order of the description. `MH.Random` specifies there is no order to enforce between any node *blocks* (we intend by block the set of all the nodes that could be generated from a unique description block like in line 25-27), except if the parent node has the `determinist` attribute. `MH.FullyRandom` specifies there is no order to enforce between every single nodes. `MH.Pick` specifies that only one node among the children should be kept at a time—the choice is randomly performed except if the parent has the `determinist` attribute—as per the weight associated to each child node (`weights`, line 55).

lines 10-14 A terminal node with typed-value contents is defined. It is a `BitField`. This node have an attribute `'qty': (1,5)` (line 14) which specifies that it can be present from 1 to 5 times. (Note that, by default, raw data absorption will also be constrained by this limit)

line 16 This pattern allows to use an already defined node. In our case, it is the node `val2` specified in lines 24-26.

lines 29-30 This pattern with the keyword `clone` allows to make a full copy of an existing node.

lines 32-34 The keywords `import_from` and `data_id` are used for importing a data type from another data model. In this case it is a `STRING Descriptor` data type from the `USB` data model.

lines 36-39 Here is defined a *generator* nodes. It takes two nodes of the current graph as parameters, namely: `(val21, 2)` and `val3`. It simply create a new node with a value equal to the contents of its node parameters.

Note: The syntax `(X, nb)`—as illustrated by `(val21, 2)`—allows to use within the description the same name `X` for different nodes having different parents while being able to reference them uniquely—thanks to `nb`—as illustrated by this generator node.

---

lines 45-50 Two alternate configurations of node `val3` are specified through this pattern.

lines 44 The keyword `sync_qty_with` allows to synchronize the number of nodes to generate or to absorb with the one specified by its name. In this case it is the node `val1` which is defined in lines 10-14.

To register such a description within the data model `MyDF` you can directly use *framework.data_model.DataModel.register()* as seen in the previous example. But if you want to access afterwards to the defined nodes, you can also transform this description to a graph, before registering it, like this:

```
1 nb = NodeBuilder(self)
2 root_node = nb.create_graph_from_desc(d1)
```

You could then access to all the registered nodes tided up in the specific dictionary `mb.node_dico`, whether you want to perform extra operation on them.

See also:

In order to make easier the modeling of data formats, some helpers are provided, namely: some *generator*-node templates (refer to *Generator Node Templates*) and some block builders (refer to *Block Builders*).

2.3.2 Visualization of Modeled Data

Let's show how to visualize an instance of the imaginary TestNode data model we just described in section *Defining the Imaginary MyDF Data Model*. It is only a matter of calling the method `.show()` on it, which will draw in ASCII what can be seen on the figure *TestNode Visualization*.

Note: You can notice that the graph paths of the modeled data are presented in a similar form as Unix file paths (for instance `TestNode/middle/val2`). As it is explained in the section *Defining Specific Disruptors*, using these paths are a typical way for referencing a node within a modeled data.

2.3.3 Absorption of Raw Data that Complies to the Data Model

2.3.3.1 A First Example

Let's begin with a simple example on how to absorb raw data that will match the imaginary TestNode data model we just described in section *Defining the Imaginary MyDF Data Model*.

```
from framework.plumbing import *

fmk = FmkPlumbing()

fmk.run_project(name="tuto")

data_gen = fmk.dm.get_atom('TestNode')    # first instance of TestNode data model
data_abs = fmk.dm.get_atom('TestNode')  # second instance of TestNode data model

raw_data = data_gen.to_bytes()
print(raw_data)
```

In our case, this code block output the following:

```
'\xc0\x027\xc0\x022@\x01\xfa\xc0\x02TOKOK\x14\x03b\x00l\x00a\x00b\x00l\x00a\x00.\x00.\
↪x00.\x00AAA.R5l%Jde==@\x02\x15.R5l%Jde==.R5l%Jde==.R5l%Jde==.R5l%Jde==PLIPAAA\x08\x04\
↪x06'
```

(Note that if you execute that on your side you will maybe get something else, as there is some random in this data model.)

And if we want to visualize it more gracefully, we can simply write `data_gen.show()` which will draw in ASCII what can be seen on the figure *TestNode Visualization*.

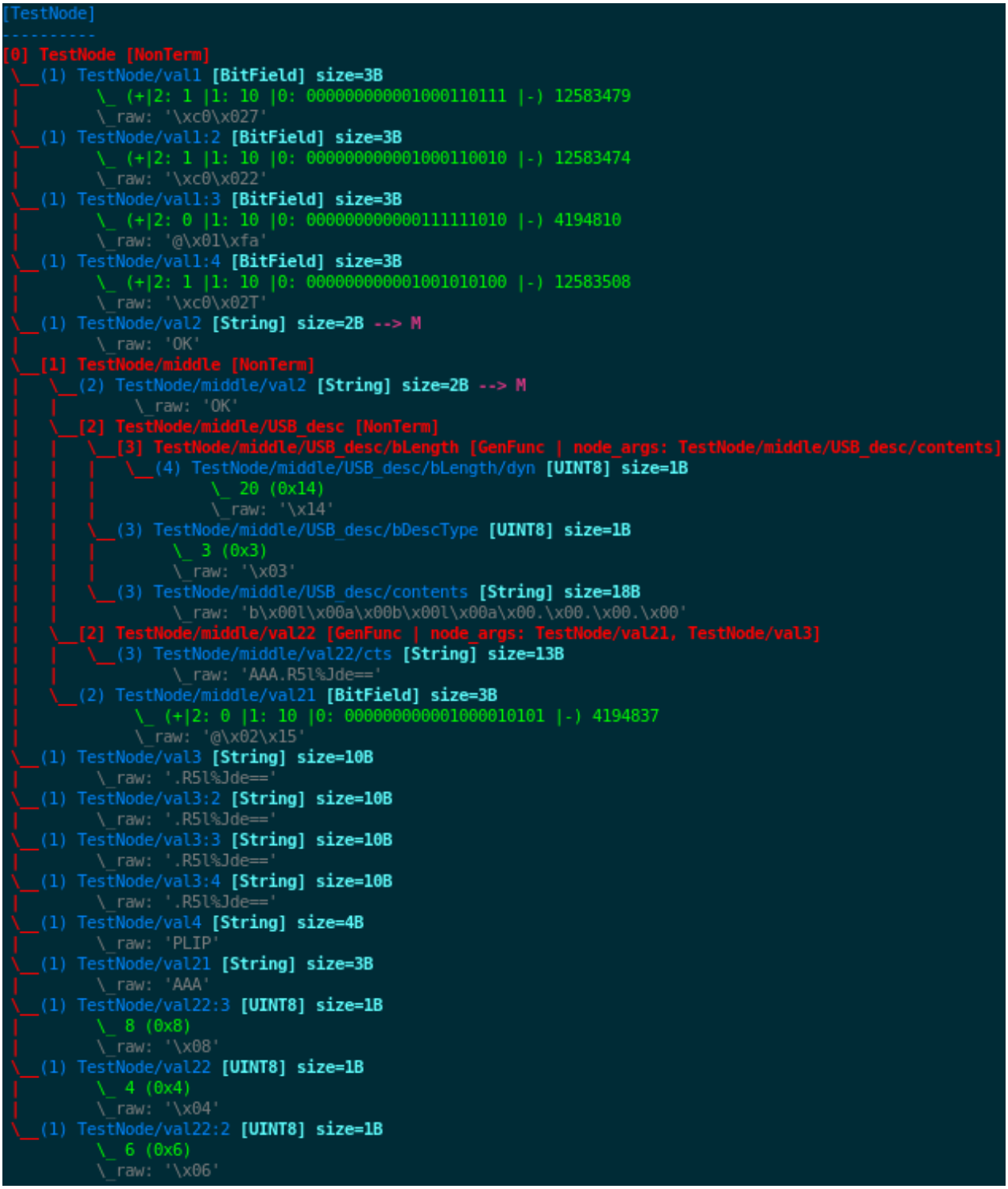


Fig. 5: TestNode Visualization

Note: You can remark that we have instanciated twice the TestNode data model in line 7 and 8. The first one referenced by `data_gen` was used to generate the previous raw data while the second one referenced by `data_abs` will be used in what follows to demonstrate absorption.

In order to absorb what have been previously generated, we will use the second data model instance `data_abs` and will call its `.absorb()` method with the previous generated data:

```
1 data_abs.absorb(raw_data)
```

The following tuple will be returned:

```
(<AbsorbStatus.FullyAbsorbed: 4>, 0, 102, 'TestNode')  # --> (status, offset, size, name)
```

The *status* is `<AbsorbStatus.FullyAbsorbed: 4>` which means that everything went well, that is, all the provided data has been absorbed. The *offset* and *size* give the part of the data that has been absorbed. In our case, it maps the full length of the original data, namely `102` bytes.

Finally, if you call the method `.show()` on the model instance `data_abs` you will see the same ASCII representation as the original one depicted by *TestNode Visualization*.

2.3.3.2 Absorption Constraints

Absorption constraints can be configured in order to accept data that does not conform completely to the defined data model, which can be helpful if this data model does not specify every aspects of a data format, or if you want to voluntarily step outside the data format requirements.

By default, when you perform an absorption, every data model constraints will be enforce. If you want to free some ones, you need to provide a `framework.node.AbsCsts` object—specifying the constraints you want—when calling the method `.absorb()`.

Currently, there is four kinds of constraints:

`size` If size matters for some nodes—for instance if `String()` size attributes are specified within a terminal node—this constraint control it.

`content` Only the values specified in the data model are accepted

`similar_content` This constraint is a lighter version of `content`. It allows values similar to the one defined in the data model to be accepted in absorption operations. This is especially leveraged by String() to distinguish case sensitive from case incensitive strings.

`regexp` This constraint control if regular expression—that some terminal nodes can specify—should be complied to.

`struct` This constraint control whether or not data structure should be complied to. That covers part of the grammar specified through non-terminal nodes: quantity of children, quantity synchronization (specified through `sync_qty_with` attribute), and existence synchronization—specified through `exists_if` or `exists_if_not` attribute.

There is also the shortcuts `framework.node.AbsNoCsts` and `framework.node.AbsFullCsts` which respectively set no constraints, or all constraints. Thus, if you want to only respect `size` and `struct` constraints, you can provide the object `AbsNoCsts(size=True,struct=True)` to the `.absorb()` method, like what follows:

```
status, off, size, name = data_abs.absorb(data, constraints=AbsNoCsts(size=True,
↪struct=True))
```

In some cases, it could also be useful to only set absorption constraints to some nodes. To do so, you can call the method *framework.node.Node.enforce_absorb_constraints()* on the related nodes with your chosen constraints. You

can also add a specific field `absorb_csts` (refer to *Data Model Keywords* and *Data Model Patterns*) within a data model description to reach the same objective.

2.3.3.3 Defining Absorption Helpers

For complex scenario of absorption, the constraints defined within the data model are not always sufficient. In such cases you could add helpers to the related nodes. Let's say you want to model something like that:

```
split_desc = \
{'name': 'test',
 'contents': [

     {'name': 'prefix',
      'contents': UINT8(values=[0xcc, 0xff, 0xee])},

     {'name': 'variable_string',
      'contents': String(max_sz=20)},

     {'name': 'keycode',
      'contents': UINT16_be(values=[0xd2d3, 0xd2fe, 0xd2aa])},

     {'name': 'variable_suffix',
      'contents': String(values=['END', 'THE_END'])}
 ]}
```

It works as intended for data generation, but if you want to absorb a data that comply to this model, you will currently need to help `fuddly` a little, as the node `variable_string` could be too greedy and absorb the `keycode` whether the raw data to absorb contains a `variable_string` strictly below the limit of the specified `20` characters, like this:

```
\xffABCDEF\xd2\xfeTHE_END
```

To help `fuddly` making the right things, you could define an helper function and associate it to the `keycode` node as illustrated in what follows:

```
def keycode_helper(blob, constraints, node_internals):
    off = blob.find(b'\xd2')
    if off > -1:
        return AbsorbStatus.Accept, off, None
    else:
        return AbsorbStatus.Reject, 0, None

split_desc = \
{'name': 'test',
 'contents': [

     {'name': 'prefix',
      'contents': UINT8(values=[0xcc, 0xff, 0xee])},

     {'name': 'variable_string',
      'contents': String(max_sz=20),
      'set_attrs': [NodeInternals.Abs_Postpone]},

     {'name': 'keycode',
```

(continues on next page)

(continued from previous page)

```
      'contents': UINT16_be(values=[0xd2d3, 0xd2fe, 0xd2aa]),
      'absorb_helper': keycode_helper},

     {'name': 'variable_suffix',
      'contents': String(values=['END', 'THE_END'])}
 ]}
```

Notice that we also add a specific attribute to the node `variable_string`, namely: `NodeInternals.Abs_Postpone`. This will instruct `fuddly` to postpone any absorption corresponding to this node, awaiting that the next node first find in the raw data what he wants. Now, if we look at the `keycode_helper()` function, we can notice that it has access to part of the raw data (the one that still need to be consumed/absorbed) through its `blob` parameter. It basically looks for a byte with the value `\xd2`. If it finds it, it will return a success status as well as the offset where it wants to start absorption (in this case it is the offset of what it finds). Note, that the last value returned in the tuple is a `size` attribute. In this case it is set to `None`, but it can enforce the size of what should be absorbed in what remains in the raw data (could be useful for instance for `String()`).

Now if you try to absorb the previous raw data, it will work as expected. This example is voluntarily simple enough to better grasp what is the purpose of having a helper. It could be legitimately expected that in this case `fuddly` do it by itself, and in fact it is currently able to do so ;) thanks to some already defined `absorb_auto_helpers` methods. Thus, in this example you could remove the *helper* stuff, while still keeping the `NodeInternals.Abs_Postpone` attribute on the node `variable_string`, and everything will work as expected.

Note: `NodeInternals.Abs_Postpone` allows to postpone the node absorption until the next node successfully absorbs part of the provided data. If this latter node fails, the postponed absorption will also fail.

See also:

The already defined auto-helper functions, behave accordingly to the typed value contents. They are more elaborated than the example *helper* function defined above. Look at the code *framework.value_types.INT.absorb_auto_helper()* and/or *framework.value_types.String.absorb_auto_helper()* in order to better understand how it works.

Even if `fuddly` can handle by itself this classic cases, you could face situations where absorption will really not be so obvious (whether you didn't put sufficient constraints within the data model, or because you don't want to for letting more freedom during data generation).

2.3.4 Describing Protocols Ruling a Data Model

Two compementary options are provided by the framework:

- The *Scenario Infrastructure* that enables you to have access to automatically-created *Generators* that comply to the protocols you described. Refer to *Scenario Infrastructure*.
- The definition of *Virtual Operators*. refer to *Defining Operators*

2.3.5 Defining Specific Disruptors

See also:

For insights on how to manipulate data, refer to *Data Manipulation*.

2.3.5.1 Overview

Specific disruptors have to be implemented within `mydf_strategy.py`. This file should starts with:

```
from framework.plumbing import *
from framework.tactics_helper import *

tactics = Tactics()
```

Note: `Fuddly` registers for each data model the related dynamically-created generators, and if defined, specific disruptors. For that purpose, an object `framework.tactics_helper.Tactics` has to be instantiated and referenced by the global variable `tactics`.

Then, to define a specific disruptor for your data model you basically have to define a subclass of `framework.tactics_helper.Disruptor` or `framework.tactics_helper.StatefulDisruptor`, and use the decorator `@disruptor` on it to register it. The first parameter of this decorator has to be the `framework.tactics_helper.Tactics` object you declare at the beginning of `mydf_strategy.py`.

```
@disruptor(tactics, dtype="DISRUPTOR_TYPE", weight=1)
class disruptor_name(Disruptor):

   def disrupt_data(self, dm, target, prev_data):

        # Do something with prev_data

        return prev_data
```

For stateful disruptor you also need to implement the method `framework.tactics_helper.StatefulDisruptor.set_seed()`. It will be called only when the disruptor needs a new data to consume. Thus, it will be called the very first time, and then each time the disruptor notify `fuddly` that it needs a new data to consume. This notification is done by calling `framework.tactics_helper.StatefulDisruptor.handover()` within `framework.tactics_helper.StatefulDisruptor.disrupt_data()`. The following code block illustrates such kind of disruptor:

```
@disruptor(tactics, dtype="DISRUPTOR_TYPE", weight=1)
class disruptor_name(StatefulDisruptor):

   def set_seed(self, prev_data):
       self.seed_node = prev_data.content

   def disrupt_data(self, dm, target, data):
       new_node = do_some_modification(self.seed_node)
       if new_node is None:
           data.make_unusable()
           self.handover()
       else:
           data.update_from(new_node)
```

(continues on next page)

(continued from previous page)

```
14            data.add_info('description of the modification')
15
16    return data
```

Note: Remark the call to the method *framework.data.Data.update_from()* (line 13). Such construction comes from the fact `fuddly` uses a data-model independent *container* (*framework.data.Data*) for passing modeled data from one sub-system to another. This container is also used, for logging purpose, to register the sequence of modifications performed on the data (especially the disruptor chain— refer to *How to Perform Automatic Modification on Data*) and other things, such as information retrieved from what a disruptor wants to report (line 14), for instance, insights on the modifications it performed.

You can also define parameters for your disruptor, by specifying the `args` attribute of the decorator with a dictionary. This dictionary references for each parameter of your disruptors a tuple composed of a description of the parameter, its default value, and the type of the value. The following example illustrates this use case, as well as the way to access the parameters within the disruptor methods.

```
1  @disruptor(tactics, dtype="DISRUPTOR_TYPE", weight=1,
2             args={'param_1': ('param_1 description', None, str),
3                   'param_2': ('param_2 description ', True, bool)})
4  class disruptor_name(StatefulDisruptor):
5
6     def set_seed(self, prev_data):
7         do_stuff(self.param_1)
8         do_other_stuff(self.param_2)
```

2.3.5.2 The Model Walker Infrastructure

The model walker infrastructure can helps you if you want to define a stateful disruptor that performs operations on the provided data, for each of its node (or for specific nodes of interest), one node at a time.

Basically, the class *framework.fuzzing_primitives.ModelWalker* takes a modeled data as a parameter and an instance of a subclass of *framework.fuzzing_primitives.NodeConsumerStub*—acting like a *visitor* but being able to modify the nodes it visits. This special *visitor* has to establish the criteria of the nodes on which it is interested in and it has to implement the method *framework.fuzzing_primitives.NodeConsumerStub.consume_node()* to perform the intended modification on such nodes.

Note: The *Model Walker* infrastructure will by default also consider the non-terminal nodes. And if the consumer is not interested on them, it will iterates on the different possible forms they can take (optional parts, various defined shapes, ...), in order for the consumer to have the opportunity to act on the different shapes the data may have.

Also, note that if you want to iterate on the different forms of a modeled data, you can use the disruptor `tWALK` with the specific parameter `nt_only` set to `True`. Refer to *Generic Disruptors*.

Let's take the following generic consumer *framework.fuzzing_primitives.SeparatorDisruption*, that replaces, one at a time, every separators of a modeled data with another inappropriate separator.

```
1  class SeparatorDisruption(NodeConsumerStub):
2
3      def init_specific(self, separators):
4          self._internals_criteria = \
```

(continues on next page)

(continued from previous page)

```
            dm.NodeInternalsCriteria(mandatory_attrs=[dm.NodeInternals.Mutable, dm.
↪NodeInternals.Separator],
                                     node_kinds=[dm.NodeInternals_Term])

        self.values = [b'']
        if separators is not None:
            self.values += list(separators)

    def consume_node(self, node):
        orig_val = node.to_bytes()
        new_values = copy.copy(self.values)

        if orig_val in new_values:
            new_values.remove(orig_val)

        node.import_value_type(value_type=vtype.String(values=new_values))
    node.unfreeze()

        node.make_finite()
        node.make_determinist()

        return True
```

In brief, at initialization, we define the kind of nodes on which we are interested in doing some operations (line 4-6). We then register the list of separator words allowed for this data. The core of our modification is implemented within the method *framework.fuzzing_primitives.SeparatorDisruption.consume_node()*, which is called by the model walker each time it encounters a node of interest, that is in our case a separator. In this method we change the separator node such as it will expand as any separator words except the legitimate one. After *framework.fuzzing_primitives.SeparatorDisruption.consume_node()* is called, the model walker will iterate over each defined shapes for this node (by issuing continuously *framework.node.Node.get_value()* then *framework.node.Node.unfreeze()*) until exhaustion or after a predefined limit.

Note: Saving and restoring the consumed nodes is performed automatically by *framework.fuzzing_primitives.NodeConsumerStub*, but depending on your needs you can override the related methods.

Finally, to make the *Model Walker* walks, you only have to instantiate it with the intended parameters, and it will return an iterator. Thus, for instance, you can display the result of the step-by-step alterations of `data_to_alter` by executing the following code snippet:

```
consumer = SeparatorDisruption()
for root_node, consumed_node, orig_val, idx in ModelWalker(data_to_alter, consumer):
    print(root_node.to_bytes())
```

If we put all things together, we can write our *separator* disruptor like this (which is a simpler version of the generic disruptor `framework.generic_data_makers.d_fuzz_separator_nodes`):

```
@disruptor(tactics, dtype="tSEP", weight=1)
class disruptor_name(StatefulDisruptor):

    def set_seed(self, prev_data):
        prev_data.content.get_value()
```

(continues on next page)

(continued from previous page)

```
        ic = dm.NodeInternalsCriteria(mandatory_attrs=[dm.NodeInternals.Separator])
        sep_list = set(map(lambda x: x.to_bytes(),
                           prev_data.content.get_reachable_nodes(internals_criteria=ic)))
        sep_list = list(sep_list)

        self.consumer = SeparatorDisruption()
        self.walker = iter(ModelWalker(prev_data.content, self.consumer))

    def disrupt_data(self, dm, target, data):
        try:
            rnode, consumed_node, orig_node_val, idx = next(self.walker)
        except StopIteration:
            data.make_unusable()
            self.handover()
            return data

        data.update_from(rnode)

        return data
```

2.3.6 Defining a Project Environment

The environment—composed of at least one target, a logger, and optionnaly some monitoring means and virtual operators—is setup within a project file located within <root of fuddly>/projects/ or within <fuddly data folder>/user_projects/. To illustrate that let's show the beginning of generic/standard_proj.py:

```
from framework.project import *
from framework.monitor import *
from framework.operator_helpers import *
from framework.plumbing import *
import framework.global_resources as gr

project = Project()
project.default_dm = ['mydf', 'zip']
# If you only want one default DM, provide its name directly as follows:
# project.default_dm = 'mydf'

logger = Logger(record_data=False, explicit_data_recording=False,
                export_raw_data=False)

printer1_tg = PrinterTarget(tmpfile_ext='.png')
printer1_tg.set_target_ip('127.0.0.1')
printer1_tg.set_printer_name('PDF')

local_tg = LocalTarget(tmpfile_ext='.png')
local_tg.set_target_path('display')

local2_tg = LocalTarget(tmpfile_ext='.pdf')
local2_tg.set_target_path('okular')

```

(continues on next page)

(continued from previous page)

```
local3_tg = LocalTarget(tmpfile_ext='.zip')
local3_tg.set_target_path('unzip')
local3_tg.set_post_args('-d ' + gr.workspace_folder)

net_tg = NetworkTarget(host='localhost', port=12345, data_semantics='TG1',
                       hold_connection=True)
net_tg.register_new_interface('localhost', 54321, (socket.AF_INET, socket.SOCK_STREAM),
                              'TG2', server_mode=True, hold_connection=True)
net_tg.add_additional_feedback_interface('localhost', 7777,
                                         (socket.AF_INET, socket.SOCK_STREAM),
                                         fbk_id='My Feedback Source', server_mode=True)
net_tg.set_timeout(fbk_timeout=5, sending_delay=3)

targets = [local_tg, local2_tg, local3_tg, printer1_tg, net_tg]
```

A project file should contain at a minimum:

- a *framework.project.Project* object (referenced by a variable `project`)
- a *framework.logger.Logger* object (*Defining the Logger*, referenced by a variable `logger`)

and optionally:

- targets (referenced by a variable `targets`, *Defining the Targets*)
- scenarios (*Scenario Infrastructure*) that can be registered into a project through the method *framework.project.Project.register_scenarios()*
- probes (*Defining Probes*)
- tasks (*Defining Tasks*)
- operators (*Defining Operators*)

A default data model or a list of data models can be added to the project through its attribute `default_dm`. `fuddly` will use this if the project is directly launched, that is either by issuing the command `run_project` in the `fuddly` shell or by using the method *framework.plumbing.FmkPlumbing.run_project()* through any `python` interpreter.

Note: An *framework.target_helpers.EmptyTarget* is automatically added by `fuddly` to any project, for dry runs. So it does not matter if you don't define a target at the beginning.

2.3.6.1 Defining the Targets

Many targets can be defined in a project file. They have to be referenced within a list pointed by the global variable `targets` of the project file.

Within the tutorial project (`projects/tuto_proj.py`), multiple targets have been defined:

- three different *framework.targets.local.LocalTarget* for interacting with local programs;
- a *framework.targets.printer.PrinterTarget* to communicate with a CUPS server;
- and finally a *framework.targets.network.NetworkTarget* that is setup with two interfaces from which data can be sent to (and feedback retrieved from), plus an additional feedback source.

Note that the network target can route data depending on their semantics (`TG1`, `TG2`) through the two created interfaces. And for data without semantics it defaults to the first interface (`TG1`).

The simplest way to play with this target is to use `netcat` as the real target. Note that some interfaces has been setup in server mode, that means that fuddly will send data when the target connects to it. The following `netcat` instances will cover our needs:

```
[term1] # nc -l 12345

[term2] # nc localhost 54321

[term3] # nc localhost 7777
```

In order to play with the routing you can use the specific data `4TG1` and `4TG2` implemented for this purpose within the data model `mydf`.

See also:

Refer to *Generic Targets* for details on the available generic targets that you can use directly or inherit from.

If you need to implement your own `Target` you have at least to inherit from *framework.target_helpers.Target* and overload the method *framework.target_helpers.Target.send_data()* which is called by `fuddly` each time data is sent to the target. Additionally, implementing *framework.target_helpers.Target.send_multiple_data()* enables to send various data simultaneously to the target. If we take the previous `NetworkTarget` example, all the registered interfaces can be stimulated at once through this method.

See also:

Other methods of *framework.target_helpers.Target* are defined to be overloaded. Look at their descriptions to learn more about what can be customized.

2.3.6.2 Defining the Logger

You should declare a *framework.logger.Logger* in your project file, and specify the parameters that make sense for your situation. The `Logger` will then be used by `fuddly` for keeping history of your interaction with the target (e.g., data sent, feedback received, ...). Statistics about data emission will also be maintained and kept in sync with the log files.

The outputs of the logger are of four types:

- `<fuddly data folder>/logs/*<project_name>_logs`: the history of your test session for the project named `project_name`. The files are prefixed with the test session starting time. A new one is created each time you run a new project or you reload the current one. Note these files are created only if the parameter `enable_file_logging` is set to True.
- `<fuddly data folder>/logs/*<project_name>_stats`: some statistics of the kind of data that has been emitted during the session. Note these files are created only if the parameter `enable_file_logging` is set to True.
- `<fuddly data folder>/exported_data/<data model name>/*.<data extension>`: the data emitted during a session are stored within the their data model directory. Each one is prefixed by the date of emission, and each one is uniquely identified within the log files.
- records within the database `<fuddly data folder>/fmkDB.db`. Every piece of information from the previous files are recorder with this database.

Some parameters allows to customize the behavior of the logger, such as:

- `record_data` which control the location of where data will be stored. If set to `False`, instead of being stored in separate files as explained previously, they will be written directly within the log files (if `enable_file_logging` is set to `True`). This parameter does not interfere with data recording within `FmkDB`.

- `explicit_data_recording`: which is used for logging outcomes further to an *framework.operator_helpers.Operator* instruction. If set to `True`, the operator would have to state explicitly if it wants the just emitted data to be recorded. Such instruction is typically used within its method *framework.operator_helpers.Operator.do_after_all()*, where the Operator can take its decision after the observation of the target feedback and/or probes outputs.
- `enable_file_logging` which is used to control the production of log files. If set to `False`, the Logger will only commit records to the `FmkDB`.

See also:

Refer to *Defining Operators* to learn more about the interaction between an Operator and the Logger.

2.3.6.3 Defining Operators

In order to automatize what a human operator could perform to interact with one or more targets, the abstracted class *framework.operator_helpers.Operator* can be inherited. The purpose of this class is to give you the opportunity to plan the operations you want to perform on the target (data type to send, type of modifications to perform on data before sending it, and so on). Thus, you could embeds all the protocol logic to be able to adapt the fuzzing strategy based on various criteria—*e.g.*, monitoring feedback, operator choices, and so on. By default, the operator is recalled after each data emission to the target, but it can also provide to `fuddly` a batch of instructions, that will be executed prior to its recall. You have also the ability to stimulate the target through its different I/O interfaces in parallel, while each of the inputs followed a specific protocol. Obviously, a monitoring infrastructure is available to support you during the decision process.

See also:

The monitoring infrastructure enables the creation of independent probes to watch or measure any kinds of parameters linked to the target or anything else. Refer to *Defining Probes* to learn how to create them.

See also:

To implement the protocol logic you should leverage the *Scenario Infrastructure*. Refer to *Scenario Infrastructure*.

If ever the *Scenario Infrastructure* does not satisfy your need, using a full-fledged state machine library such as toysm should do.

To define an operator you have to define a class that inherits from *framework.operator_helpers.Operator*. Then, to register it within your project, the decorator `@operator` has to be used with at least the reference of the project as the first parameter.

See also:

Parameters can be defined for an operator, in order to make it more customizable. The way to describe them is the same as for *disruptors*. Look into the file `projects/generic/standard_proj.py` for some examples.

Here under is presented a skeleton of an Operator:

```
1  @operator(project)
2  class MyOperator(Operator):
3
4      def start(self, fmk_ops, dm, monitor, target, logger, user_input):
5          # Do some initialization stuff
6          return True
7
8      def stop(self, fmk_ops, dm, monitor, target, logger):
9          # Do some termination stuff
10
11     def plan_next_operation(self, fmk_ops, dm, monitor, target, logger, fmk_feedback):
```

(continues on next page)

(continued from previous page)

```
        op = Operation()

        # Do some planning stuff and decide what would be the next
        # operations you want fuddly to perform

        return op

    def do_after_all(self, fmk_ops, dm, monitor, target, logger):
        linst = LastInstruction()

        # Do some stuff after the planned Operation() has been
        # executed and request fuddly to perform some last-minute
        # instructions.

        return linst
```

The methods *framework.operator_helpers.Operator.start()* and *framework.operator_helpers.Operator.stop()* are the obvious ones that you have to implement if you want to customize the initialization and termination of your operator.

The core of your operator will be implemented within the method *framework.operator_helpers.Operator.plan_next_operation()* which will order `fuddly` to perform some operations based on the *framework.operator_helpers.Operation()* object that you will return to it. A basic example illustrating the implementation of this method is given here under:

```
def plan_next_operation(self, fmk_ops, dm, monitor, target, logger, fmk_feedback):
    op = Operation()

    if fmk_feedback.is_flag_set(FmkFeedback.NeedChange):
       op.set_flag(Operation.Stop)
    else:
       actions = [('SEPARATOR', UI(determinist=True)), ('tSTRUCT', UI(deep=True))]
       op.add_instruction(actions)

    return op
```

We instruct `fuddly` to execute a *disruptor chain* made of the `SEPARATOR` *generator* (transparently created by `fuddly` from the eponymous data type in the data model `mydf`) and the `tSTRUCT` *disruptor* with some parameters (given through `framework.tactics_helpers.UI`). And we handle the case when the *chain* has been drained. More precisely, we decide to give up when `fuddly` inform us that the stateful disruptor `tSTRUCT` has fully consumed its input, and cannot provide more outputs without re-enabling a previous stateful disruptor or in our case the *generator* from the chain.

See also:

refer to *How to Perform Automatic Modification on Data* for information about *disruptor chains*. And refer to *Defining Specific Disruptors* for insight into disruptors.

Finally, the method *framework.operator_helpers.Operator.do_after_all()* is executed by `fuddly` after the planned operation has been handled, in order for the operator to provide some last-minute instructions related to the previous operation. Typically, it is the moment where the operator can investigate on the impact of its last operation, before going on with the next one. An example leveraging this method is discussed in the following section *Defining Probes*.

Note: The methods described in this section come with some useful parameters provided by `fuddly` when it calls

them:

- `fmk_ops`: an object that exports `fuddly`'s specific methods to the operator, more precisely it is a reference to *framework.plumbing.ExportableFMKOps*.
- `dm`: a reference to the current *framework.data_model.DataModel*.
- `monitor`: a reference to the monitor subsystem, in order to start/stop probes and get status from them.
- `target`: a reference to the current target.
- `logger`: a reference to the logger.
- `fmk_feedback`: an object that provides feedback from `fuddly` to the operator about the last operation it performed. The class of this object is `framework.plumbing.FmkFeedback`.

2.3.6.4 Defining Probes

Probes are special objects that have to implement the method *framework.monitor.Probe.main()* which is called either continuously (the basic *probe*) or after a specific event in the sending process (the *blocking probes*). In order to be started, they have to be first associated to one or more *framework.target_helpers.Target* of the project. Then, when such a target is started, `fuddly` take care of running the probes.

Probes are executed independently from each other (they run within their own thread). They can interact with the target, and also use the logger. Any usage matching your expectation should be fine. Their purpose is to help you getting feedback from the target you interact with, but they can also be part of the interaction if that seems useful in your setup.

Depending on the kind of probes you want, you will have to choose between two decorators:

- `@probe` for basic probes which run continuously once started. Note there is a delay between each call to *framework.monitor.Probe.main()* which is configurable.
- `@blocking_probe` for probe which will be run just once after each data emission (default) or after each target feedback retrieval. The default behaviour can be changed by giving a `after_target_feedback_retrieval` parameter set to `True`.

These *decorators* have to take the reference of the project as parameter, in order to register them within. A really basic example (not really useful ;) of a basic probe is presented below:

```
@probe(project)
class my_first_probe(Probe):

    def start(self, dm, target, logger):
        self.cpt = 10

    def main(self, dm, target, logger):
        self.cpt -= 1
        return ProbeStatus(self.cpt)
```

A more useful one (a *blocking probe* in this case) that tries to get information from the target is given here under:

```
@blocking_probe(project, after_target_feedback_retrieval=False)
class health_check(Probe):

    def start(self, dm, target, logger):
        self.pstatus = ProbeStatus(0)
```

(continues on next page)

(continued from previous page)

```
    def stop(self, target, logger):
        pass

    def main(self, dm, target, logger):

        def check(target):
            status = 0
            # Do some monitoring of the target
            return status

        status_code = check(target)

        self.pstatus.value = status_code

        if status_code < 0:
            self.status.set_private_info("Something is wrong with the target!")

        return self.pstatus
```

Note: You can implement *framework.monitor.Probe.start()* and/or *framework.monitor.Probe.stop()* methods if you need to do some stuff during their initialization and termination.

The return status of a probe has to comply with some rules in order to get `fuddly` handle status as expected. Status rules are described below:

- If the status is positive or null, `fuddly` will consider that the target is OK.
- If the status is negative, `fuddly` will consider that something happen to the target and will act accordingly by:
  1. logging feedback from the probes as well as the status they return to facilitate further investigation;
  2. trying to recover the target, by calling *framework.target_helpers.Target.recover_target()*.

To quickly retrieve the data that negatively impacted a target and which have been recorded within the FmkDB (refer to *Defining the Logger*) you can run `tools/fmkdb.py --data-with-impact -v`. It will display for each target the data you sent for which a negative status has been recorded, coming either from:

- a probe;
- an operator (more about that in what follows);
- or the *framework.target_helpers.Target* itself (refer to the error status that are transmitted by the generic targets—*Generic Targets*).

See also:

Refer to *Generic Probes and Backend* for details on the available generic probes that you can use within your project.

In order to associate one or more probe to a target, you have to add them within the `targets` global variable of the related project file (refer to *Defining a Project Environment*). More precisely, for a target `A`, instead of putting it directly within the `targets` list, you have to put a tuple containing first the target itself, then all the needed probes. Here under an example with the target `A` associated to the probe `health_check`, and the target `B` with no probe:

```
targets = [(A, health_check), B]
```

You can use any number of probes with any target, and use the same probes for several targets. Moreover, if you want to specify a specific delay for a basic probe, you can do it by replacing the probe within `targets` with a tuple containing the probe itself and the delay expressed in seconds. Here under an example:

```
targets = [ (A, health_check, (my_first_probe, 1.4)),
            (B, (my_first_probe, 0.6)) ]
```

Finally, you can also leverage probes from within an Operator. If you want to get a status from probes each time your planned operations have been executed by `fuddly`, you can do it within the method *framework.operator_helpers.Operator.do_after_all()*. Let's illustrate this with the following example:

```
class MyOperator(Operator):

    def start(self, fmk_ops, dm, monitor, target, logger, user_input):
        if not monitor.is_probe_launched(health_check):
            # This case occurs if the probe is not associated to the target
            monitor.start_probe(health_check)

    def stop(self, fmk_ops, dm, monitor, target, logger):
        monitor.stop_probe(health_check)

    def plan_next_operation(self, fmk_ops, dm, monitor, target, logger, fmk_feedback):
        self.op = Operation()

        # Let's say the actions to be performed
        # are guided by a state machine
        self.op_state = ... # save the current state of the operator

        return self.op

    def do_after_all(self, fmk_ops, dm, monitor, target, logger):
         linst = LastInstruction()

         health_status = monitor.get_probe_status(health_check)

         if health_status.value < 0 and self.op_state == 'critical':
             linst.set_instruction(LastInstruction.RecordData)
             linst.set_operator_feedback('Data sent seems worthwhile!')
             linst.set_operator_status(-3)

         return linst
```

In this example, the operator retrieves the status of our *health-check* probe and also check what was just performed. It then correlates both information in order to determine if the test case is worth to investigate further. In our example, it occurs when the *health check* is negative and our operator state is `'critical'`. In such situation, we first order `fuddly` to record the data (line 26).

Note: In the case the logger has its parameter `explicit_data_recording` set to `True` (refer to *Defining the Logger*), you have to instruct explicitly `fuddly` to do it if you want to keep the data, otherwise it will never be logged.

Finally we convey the operator verdict to `fuddly` through the *framework.operator_helpers.LastInstruction* object it returns, by setting a negative status and some feedback on it.

Note: Setting a negative status through *framework.operator_helpers.LastInstruction* will make `fuddly` act the same as for a negative status from a probe. In addition, the operator will be shutdown.

2.3.6.5 Defining Tasks

Contrary to probes (*Defining Probes*), Tasks are not sequenced by the framework, they run asynchronously. They can be periodic or one-shot and their logic need to be defined entirely by the user. They can be started either when a target is launched (see below) or by a step of a scenario (refer to *Steps*).

To implement the logic of the task, you need to inherit from *libs.utils.Task* and to implement the `__call__()` method. This method is then called either once or with a period that is specified in the constructor. When run by the framework this task has some attributes automatically filled that you can leverage in your logic:

- *libs.utils.Task.feedback_gate*: provide an access to the last 10 seconds of feedback. (*framework.database.FeedbackGate*)
- *libs.utils.Task.dm*: current loaded data model.
- *libs.utils.Task.targets*: enabled targets.
- *libs.utils.Task.fmkops*: provide access to some framework operations (*framework.plumbing.ExportableFMKOps*).

Moreover, you could also print some information in another terminal window dedicated to the task. For such case, you should set the parameter `new_window` of the *libs.utils.Task* constructor to `True`, then use a specific API composed of *libs.utils.Task.print()* and *libs.utils.Task.print_nl()*.

Like with probes (*Defining Probes*), you can associate tasks to `targets` in order to execute them when a target is enabled. But they need to be instantiated first (while probes are only referenced by class name). Thus, for instance, if you want to run one or more tasks when a target `A` is enabled by the framework you have to put in the project `targets` list, a tuple containing first the target itself, then all the needed instantiated tasks.

Here under an example with the target `A` associated to a task of class name `myTask`, and the target `B` without tasks:

```
1 targets = [(A, myTask()), B]
```

Note: You can mix probes and tasks

CHAPTER

THREE

DATA MODELING

3.1 Data Model Keywords

This section describe the *keywords* that you could use within the frame of the *framework.node_builder.NodeBuilder* infrastructure. This infrastructure enables you to describe a data format in a JSON-like fashion, and will automatically translate this description to `fuddly`'s internal data representation.

3.1.1 Generic Description Keywords

name Within `fuddly`'s data model every node has a name that should be unique only within its siblings. But when it comes to use the *framework.node_builder.NodeBuilder* infrastructure to describe your data format, if you want to use the same name in a data model description, you have to add an extra key to keep it unique within the description, and thus allowing you to refer to this node anywhere in the description. The following example result in giving the same name to different nodes:

```
'my_name'
('my_name', 'namespace_1')
('my_name', 'namespace_2')
```

These names serve as *node references* during data description. Another option is to use the keyword `namespace`.

Note: The character `/` is reserved and shall not be used in a *name*.

---

namespace [For non-terminal nodes only]

Specify a namespace that will be used for the `name` of all the nodes reachable from the node declaring the namespace. It means that the subnodes will be automatically described by a tuple with the namespace value as second item (refer to the description of the keyword `name`).

from_namespace If specified, and some node name are used as inputs (refer for instance to keywords `clone` or `node_args`), the specified namespace will be used to fetch the nodes. It is equivalent to use a tuple expression for referring to the node with the `from_namespace` value as second item.

Note that referring to a node without specifying the namespace will be interpreted as using the current namespace. By default (even when no namespaces have been declared), one namespace is always defined from the root node, it can be referred to by `NodeBuilder.RootNS`.

contents Every node description has at least a `name` and a `contents` attributes (except if you refer to an already existing node, and in this case you have to use only the name attribute with the targeted node reference). The type of the node you describe will directly depends on what you provide in this field:

- a python `list` will be considered as a non-terminal node;

- a *Value Type* (refer to *Value Types*) will define a terminal node
- a python `function` (or everything with a `__call__` method) will be considered as a generator.
- a *framework.node.Node* will be used as a baseline for the description. If no additional keyword is provided, the provided node will be used as is. Otherwise, the additional keywords will be used to complement the description. Note that the *keyword* `name` should not be provided as it will be picked from the provided node.
- a python `regular expression` will represent a node that is terminal or non-terminal but only contains terminal ones (refer to *How to Describe a Data Format That Contains Complex Strings*).

Note that for defining a *function node* and not a generator node, you have to state the type attribute to `MH.Leaf`.

default Default value for the node. Only compatible with typed nodes (*framework.node.NodeInternals_TypedValue*). It is directly linked to the `default` parameter of each type constructor. Refer to *Value Types* for more information.

description Textual description of the node. Note this information is shown by the method *framework.node.Node.show()*.

qty Specify the amount of nodes to generate from the description, or a tuple `(min, max)` specifying the minimum (which can be 0) and the maximum of node instances you want `fuddly` to generate.

Note `-1` means infinity. It makes only sense for absorption operation (refer to *Absorption of Raw Data that Complies to the Data Model*), because for data generation, a strict limit (*framework.node.NodeInternals_NonTerm.INFINITY_LIMIT*) is set to avoid getting unintended too big data. If you intend to get such kind of data, specify explicitly the maximum, or use a disruptor to do so (*Defining Specific Disruptors*).

default_qty Specify the default amount of nodes to generate from the description. It should be within `<min, max>`.

clone Allows to make a full copy of an existing node by providing its reference.

type Used only by the *framework.node_builder.NodeBuilder* infrastructure if there is an ambiguity to determine the node type. This attributes accept the following values:

- `MH.Leaf`: to specify a terminal node, either a *value type* or a *function*.
- `MH.NonTerminal`: to specify a *non terminal* node.
- `MH.Generator`: to specify a *generator* node.

alt Allows to specify alternative contents, by providing a list of descriptors like here under:

```
'alt': [ {'conf': 'config_n1',
          'contents': SINT8(values=[1,4,8])},
         {'conf': 'config_n2',
          'contents': UINT16_be(min=0xeeee, max=0xff56),
          'determinist': True} ]
```

conf Used within the scope of the description of an alternative configuration. It set the name of the alternative configuration.

evolution_func This attribute allows to provide a function that will be used in the case the described node is instantiated more than once by a containing non-terminal node further to a *framework.node.Node.freeze()* operation (refer to the `qty` keyword). The function will be called on every node instance (but the first one) before this node incorporate the frozen form of the non-terminal. Besides, the node returned by the function will be used as the base node for the next instantiation (which makes node evolution easier). The function shall have the following signature:

```
func_name( Node ) --> Node
```

custo_set, custo_clear These attributes are used to customize the behavior of the described node. `custo_set` is to enable some behavior modes, whereas `custo_clear` allows to disable them. What is expected is either a single mode or a list of modes. The available modes depend on the kind of node.

For non-terminal node, the customizable behavior modes are:

- `MH.Custo.NTerm.MutableClone`: By default, this mode is *enabled*. When enabled, it means that for child nodes which can be instantiated many times (refer to `qty` attribute), all instances will be set as *mutable*. If it is disabled, when a child node is instantiated more than once, only the first instance is set *mutable*, the others have this attribute cleared to prevent generic disruptors from altering them. This mode aims at limiting the number of test cases, by pruning what is assumed to be redundant.
- `MH.Custo.NTerm.CycleClone`: By default, this mode is *disabled*. When enabled, and when the subnodes need to be duplicated because of a `qty` greater than 1, the non-terminal node will walk through each copy, in order to cycle among the various shapes/values of the subnodes. Note this customization won't be effective if an evolution function is provided through the keyword `evolution_func`.
- `MH.Custo.NTerm.FrozenCopy`: By default, this mode is *enabled*. When enabled, it means that for child nodes which can be instantiated many times (refer to `qty` attribute), the instantiation process will make a frozen copy of the node, meaning that it will be the exact copy of the original one at the time of the copy. If disabled, the instantiation process will ignore the frozen state, and thus will release all the constraints.
- `MH.Custo.NTerm.FullCombinatory`: By default, this mode is *disabled*. When enabled, walking through a non-terminal node will generate all "possible" combination of forms for each subnode. The various considered forms for a subnode are based on the `qty` and `default_qty` parameter provided. Thus there are at most 3 different forms that boil down to the different amounts of subnodes (max, min and default values), and at least 1 if all are the same. Other possible values in the range `<min, max>` are reachable in `random` mode, or by changing the subnode quantity manually. When this mode is disabled, walking through the non-terminal node won't generate all possible combinations but a subset of it based on a simpler algorithm that will walk through each subnode and iterate for their different forms without considering the previous subnodes forms.

  Note: Note that if the node is not frozen at the time of the copy, this customization won't have any effect. The main interest is in conjunction with the *disruptors* (like `tTYPE`, `tWALK`, ...) which are based on the `ModelWalker` infrastructure (refer to *The Model Walker Infrastructure*). Indeed, this infrastructure releases constraints on non-terminal nodes before providing a new model instance. Releasing constraints triggers child nodes reconstruction for each non-terminal. And as the terminal children will be frozen at that time, the reconstruction will take into account this customization mode.

- `MH.Custo.NTerm.StickToDefault`: By default, this mode is *disabled*. When enabled, walking through a non-terminal node *won't* generate all "possible" combination of forms for each subnode. Only the default quantity (refer to keyword `default_qty`) is leveraged. Walking through such nodes will generate new forms only if different shapes have been defined (refer to keyword `shape_type` and `section_type`).
- `MH.Custo.NTerm.CollapsePadding`: By default, this mode is *disabled*. When enabled, every time two adjacent `BitField` 's (within its scope) are found, they will be merged in order to remove any padding in-between. This is done "recursively" until any inner padding is removed.

  Note: To be compatible with an *absorption* operation, the non-terminal set with this customization should comply with the following requirements:

  - The `lsb_padding` parameter shall be set to `True` on every related `BitField` 's.
  - The `endian` parameter shall be set to `VT.BigEndian` on every related `BitField` 's.

  - the `qty` keyword should not be used on the children except if it is equal to `1`, or `(1,1)`.

- `MH.Custo.NTerm.DelayCollapsing`: By default, this mode is *disabled*. To be used in conjunction with `MH.Custo.NTerm.CollapsePadding` when the collapse operation should not be performed in the current non-terminal node but in the parent node. Refer to the code snippet below for an example:

```
{'name': 'request',
 'custo_set': MH.Custo.NTerm.CollapsePadding,
 'contents': [
    {'name': 'header',
     'contents': BitField(subfield_sizes=[3,1], endian=VT.BigEndian,
                          subfield_val_extremums=[[0,7], [0,1]])},

    {'name': 'payload',
     'custo_set': [MH.Custo.NTerm.CollapsePadding, MH.Custo.NTerm.
↪DelayCollapsing],
     'contents': [
         {'name': 'status',
          'contents': BitField(subfield_sizes=[1,3], endian=VT.BigEndian,
                               subfield_values=[None,[0,1,2]])},
         {'name': 'count',
          'contents': UINT16_be()}
      ]},

      # [...]
  }
```

  Without this mode, when resolving the *request* node to get the byte-string the *payload* subnode will be resolved too early and will produce a byte-string without any collapse operation.

For *generator* node, the customizable behavior modes are:

- `MH.Custo.Gen.ForwardConfChange`: By default, this mode is *enabled*. If enabled, a call to *framework.node.Node.set_current_conf()* will be called on the generated node (default behavior).
- `MH.Custo.Gen.CloneExtNodeArgs`: By default, this mode is *disabled*. If enabled, during a cloning operation (e.g., full copy of the modeled data containing this node) if the node parameters do not belong to the graph representing the data, they will be cloned (full copy). Otherwise, they will just be referenced (default behavior). Rationale for default behavior: When a *generator* or *function* node is duplicated within a non terminal node, the node parameters may be unknown to it, thus considered as external, while still belonging to the full data.
- `MH.Custo.Gen.ResetOnUnfreeze`: By default, this mode is *enabled*. If enabled, a call to *framework.node.Node.unfreeze()* on the node will provoke the reset of the *generator* itself, meaning that the next time its value will be asked for, it will be recomputed (default behaviour). If unset, a call to the method *framework.node.Node.unfreeze()* will provoke the call of this method on the already existing generated node (and if it didn't exist by this time it would have been computed first).
- `MH.Custo.Gen.TriggerLast`: By default, this mode is *disabled*. If enabled, the triggering of a generator is postpone until everything else has been resolved. It is especially useful when you describe a generator that use a node with an existence condition and that this condition cannot be resolved at the time the generator would normally trigger (which is when it is reached while walking through the graph).

For *function* node, the customizable behaviors mode are:

- `MH.Custo.Func.FrozenArgs`: By default, this mode is *enabled*. When enabled, the node parameters are frozen before being provided to the *function* node. If disabled, the node parameters are directly provided

to the *function* node (without being frozen first).

- `MH.Custo.Func.CloneExtNodeArgs`: By default, this mode is *disabled*. Refer to the description of the corresponding *generator node* mode.

3.1.2 Keywords to Describe Non Terminal Node

shape_type Allows to choose the order to be enforce by a non-terminal node to its children. `MH.Ordered` specifies that the children should be kept strictly in the order of the description. `MH.Random` specifies there is no order to enforce between any *node descriptor* (which can expand to several nodes), except if the parent node has the `determinist` attribute. `MH.FullyRandom` specifies there is no order to enforce between every single nodes. `MH.Pick` specifies that only one node among the children should be kept at a time—the choice is randomly performed except if the parent has the `determinist` attribute—as per the weight associated to each child node.

weight Used within the scope of a shape description for a non-terminal node. A non-terminal node can organize all its child nodes in various way by describing different shapes. Each shape has a weight which is used either—when the non-terminal node is random—as a way to determine the chance that `fuddly` we use it during the data generation process, or as a mean to order the shape—when the node is put in determinist mode. Let's look at the example here under:

```
{'name': 'test',
 'contents': [

      # SHAPE 1
      {'weight': 20,
       'contents': [
           {'section_type': MH.Random,
            'contents': [
                {'contents': String(max_sz=10),
                 'name': 'val1',
                 'qty': (1, 5)},

       ...

      # SHAPE 2
      {'weight': 10,
       'contents': [
           {'section_type': MH.FullyRandom,
            'contents': [
                {'name': 'val1'},

       ...
```

Note: A *shape description* is composed of the two attributes `weight` and `contents`.

section_type Similar to `shape_type` keyword. But only valid for describing a section within a non-terminal node, and limited to this section. The following example illustrates that:

```
{'name': 'test',
 'shape_type': MH.Random
 'contents': [
```

(continues on next page)

(continued from previous page)

```
        {'name': 'val1',
         'contents': String(values=['OK', 'KO']),
         'qty': (0, 5)},

        {'section_type': MH.Ordered,
         'contents': [

                {'name': 'val2',
                 'contents': UINT16_be(values=[10, 20, 30])},

                {'name': 'val3',
                 'contents': String(min_sz=2, max_sz=10, alphabet='XYZ')},

                {'name': 'val4',
                 'contents': UINT32_le(values=[0xDEAD, 0xBEEF])},

         ]}

        {'name': 'val5',
         'contents': String(values=['OPEN', 'CLOSE']),
         'qty': 3}
]}
```

duplicate_mode Modify the behavior of the instantiating procedure when a child node is instantiated more than once. This can be set to:

- `MH.Copy`: A new instance corresponds to a full copy operation.
- `MH.ZeroCopy`: A new instance corresponds to a new reference of the child node.

weights To be used optionally in the frame of a non-terminal node along with a `MH.Pick` type. If used this attribute shall contains an integer tuple describing the weight for each one of the subsequent nodes to be picked. Can be used within a section description, or directly in the non-terminal nodes, if it has a `MH.Pick` type.

separator When specified, the non-terminal will add a separator between each one of its children. This attribute has to be filled with a *separator descriptor* such as what is illustrated below:

```
'separator': {'contents': {'name': 'sep',
                           'contents': String(values=['\n'])},
              'prefix': False,
              'suffix': False,
              'unique': True,
              'always': False},
```

The keys `prefix`, `suffix`, `unique` and `always` are optional. They are described below.

See also:

Refer to *How to Describe the Separators of a Data Format* for an example using separators.

prefix Used optionally within a *separator descriptor*. If set to `True`, a separator will be placed just before the first child.

suffix Used optionally within a *separator descriptor*. If set to `True`, a separator will be placed just after the last child.

unique Used optionally within a *separator descriptor*. If set to `True`, the inserted separators will be independent from each other (full node copy). Otherwise, the separators will be references to a unique node (zero copy).

always Used optionally within a *separator descriptor*. If set to `True`, the separator will be always generated even if the subnodes it separates are not generated because their evaluated quantity is 0.

encoder If specified, an encoder instance should be provided. The *encoding* will be applied transparently when the binary value of the non terminal node will be retrieved (`framework.node.Node.to_bytes()`). Additionally, during an absorption (refer to *Absorption of Raw Data that Complies to the Data Model*), the *decoding* will also be performed automatically.

Several generic encoders are defined within `framework/encoders.py`. But if they don't match your need, you can define your own encoder by inheriting from `framework.encoders.Encoder` and implementing its interface.

See also:

Refer to *How to Describe a Data Format With Some Encoded Parts* for an example on how to use this keyword.

Note: Depending on your needs, you could also choose to implement a disruptor to perform your encoding (refer to *Defining Specific Disruptors*).

3.1.3 Keywords to Describe Generator Node

node_args List of node parameters to be provided to a *generator* node or a *function* node.

other_args List of parameters (which are not a `framework.node.Node`) to be provided to a *generator* node or a *function* node.

provide_helpers (Optional) If set to *True*, a special object will be provided to the user-defined function (last parameter) of the *generator* node or the *function* node. Otherwise, this object won't be passed (default behavior). This object is an instance of the class `framework.node.DynNode_Helpers`, which enable the user-defined function to have some insight on the current structure of the modeled data.

trigger_last This keyword is a shortcut for the related node customization mode. Refer to `custo_set` and `custo_clear`.

3.1.4 Keywords to Import External Data Description

import_from Name of the data model to import a data description from.

data_id Name of the data description to import.

3.1.5 Keywords to Describe Node Properties

determinist Make the node behave in a deterministic way.

random Make the node behave in a random way.

finite Make the node *finite*, meaning that it will exhaust at some point (meaning that it has cycled over all its possible values or shapes) When the situation occurs, a notification is posted in the node environment (refer to *Data Manipulation*)

infinite Make the node *infinite*, meaning that it will always provide values.

mutable Make the node mutable. It is a shortcut for the node attribute `MH.Attr.Mutable`.

highlight Make the node highlighted. It is a shortcut for the node attribute `MH.Attr.Highlight`.

set_attrs List of attributes to set on the node. The current generic attributes are:

- `MH.Attr.Freezable`: If set, the node will be freezable (default behavior), which means that once the node has provided a value (through for instance *framework.node.Node.to_bytes()*), the method *framework.node.Node.unfreeze()* need to be called on it to get new values, otherwise it won't change. If unset, the node will always be recomputed. Can be useful for *function* node, if it needs to be recomputed each time a modification has been performed on its associated graph (e.g., CRC function).
- `MH.Attr.Mutable`: If set, generic disruptors will consider the node as being mutable, meaning that it can be altered (default behavior). Otherwise, it will be ignored. When a non-terminal node has this attribute, generic disruptors using the ModelWalker algorithm (like `tWALK` and `tTYPE`) will stick to its default form (meaning default quantity will be used for each subnodes and if the node has multiple shapes, the higher weighted one will be used. Likewise for *Pick* sections). Also, the method *framework.node.Node.unfreeze()* won't perform any changes on non-terminal nodes which are not mutable.
- `MH.Attr.Determinist`: This attribute can be set directly through the keywords `determinist` or `random`. Refer to them for details. By default, it is set.
- `MH.Attr.Finite`: If set, a node will provide a finite number of values and then will notify it has exhausted. Otherwise, exhaustion will never be notified (default behavior).
- `MH.Attr.Abs_Postpone`: Used to postpone absorption by the node. Refer to *Absorption of Raw Data that Complies to the Data Model* for more information on that topic.
- `MH.Attr.Separator`: Used to distinguish a separator. Some disruptors can leverage this attribute to perform their alteration.
- `MH.Attr.Highlight`: If set, make the framework display the node in color when printed on the console. This attribute is also used by some disruptors to show the location of their modification.

Note: Most of the generic stateful disruptors will recursively set the attributes `MH.Attr.Determinist` and `MH.Attr.Finite` on the provided data before performing any alteration.

Note: *Generator* node will transfer the generic attributes to the generated node, except for `MH.Attr.Freezable`, and `MH.Attr.Mutable` which are used to change the *generator* behavior. (If such attributes need to be set or cleared on the generated node, it has to be done directly on it and not on its generator.) Specific attributes related to generators won't be passed to the generated node.

See also:

The attributes are defined within *framework.node.NodeInternals*.

clear_attrs List of attributes to clear on the node. The current attributes are the same than for the `set_attrs` keyword.

absorb_csts Used to specify some absorption constraints on the node. Refer to *Absorption of Raw Data that Complies to the Data Model* for more information on that topic.

absorb_helper Used to specify an absorption helper function for the node. Refer to *Absorption of Raw Data that Complies to the Data Model* for more information on that topic.

semantics Used to specify semantics to the node, by way of a list of meaningful strings. Nodes can be searched for and selected based on semantics. Refer to *Data Manipulation* for more information on that topic.

fuzz_weight Used by some stateful disruptors to order their test cases. The heavier the weight, the higher the priority of handling the node.

sync_qty_with Allow to synchronize the number of node instances to generate or to absorb with the one specified by reference.

qty_from Allow to synchronize the number of node instances to generate or to absorb with the *value* of the one specified by reference. You can also specify an optional *base quantity* that will be added to the retrieved value. In this case, you shall provide a `list/tuple` with first the node reference then the *base quantity*.

This keyword is the counterpart of the *generator template* *framework.dmhelpers.generic.QTY*. It is preferable to this *generator* when the node from which the quantity is retrieved is already resolved at retrieval time. In this case *generation* and *absorption* operations will be handled transparently.

sync_size_with, sync_enc_size_with Allow to synchronize the length of the described node (the one where this keyword is used) with the *value* of the node specified by reference (which should be an *framework.value_types.INT*-based typed-node). These keywords are useful for size-variable node types. They are currently supported for typed-nodes which are *framework.value_types.String*-based with or without an encoding. Non-terminal nodes are not supported (for absorption). The distinction between `sync_size_with` and `sync_enc_size_with` is that the synchronization will be performed:

- either with respect to the length of the data retrieved from the node in a *decoded* form. *Decoded* means that it is agnostic to the *codec* specified (e.g., `utf-8`, `latin-1`, ...) in the `String`, and also, for `Encoded-String` (e.g., *framework.value_types.GZIP*, ...) , that it is agnostic to any *framework.encoders.Encoder* the `String` is wrapped with;
- or with respect to the length of the encoded form of the data.

Generation and absorption deal with these keywords differently, in order to achieve the expected behavior. For generation, the synchronization goes from the described node to the referenced node (meaning that the data is first pulled from the size-variable node, then the referenced node is set with the length of the pulled data). Whereas for the absorption it goes the other way around.

Note also that you can provide an optional *base size* that will be added to the length before synchronization in the case of generation, and removed from the length in the case of absorption. In this case, you shall provide a `list/tuple` with first the node reference then the *base size*.

These keywords are the counterpart of the *generator template* *framework.dmhelpers.generic.LEN*. They are preferable to this *generator* (when the size-variable node is not a non-terminal), because *generation* and *absorption* operations will be handled transparently thanks to them.

exists_if Enable to determine the existence of this node based on a given condition.

See also:

Refer to *How to Describe a Data Format Whose Parts Change Depending on Some Fields* for how to use existence conditions.

exists_if/and, exists_if/or Extend the `exists_if` keyword by allowing to specify a list or a tuple of conditions. The operator `and` (respectively `or`) will be used to generate the desired behaviour.

```
{'name': 'test',
 'contents': [
    {'name': 'opcode',
     'contents': String(values=['A3', 'A2'])},
    {'name': 'subopcode',
     'contents': BitField(subfield_sizes=[15,2,4],
                          subfield_values=[[500], [1,2], [5,6,12]])},
    {'name': 'and_condition',
     'exists_if/and': [(RawCondition('A2'), 'opcode'),
                       (BitFieldCondition(sf=2, val=[5]), 'subopcode')],
     'contents': String(values=['and_condition_true'])}
 ]}
```

exists_if_not Enable to determine the existence of this node based on the non-existence of another one.

post_freeze To be filled with a function. If specified, the function will be called just after the node has been frozen. It takes the node internals as argument (*framework.node.NodeInternals*).

specific_fuzzy_vals Usable for *typed-nodes* only. This keyword allows to specify a list of additional values to be leveraged by the *disruptor* `tTYPE` (*tTYPE - Advanced Alteration of Terminal Typed Node*) while dealing with the related node. These additional values are added to the test cases planned by the *disruptor* (if not already planned).

charset Used in the context of a *regular expression* `contents`. It enables to specify the charset that will be considered for interpreting the regular expression and for creating the related nodes. Accepted attributes are:

- `MH.Charset.ASCII`
- `MH.Charset.ASCII_EXT` (default)
- `MH.Charset.UNICODE`

3.1.6 Keywords to Describe Constraints

constraints List of node constraints specified through *framework.constraint_helpers.Constraint* objects. They will be added to a CSP (Constraint Satisfiability Problem) associated to the currently described data, and resolved when `Node.freeze()` is called with the parameter `resolve_csp` set to True (this is performed by default by the operator `tWALK`). It should always be associated to a non-terminal node. Refer to *How to Describe Constraints of Data Formats* for details on how to leverage such feature.

Specific operators have been defined to handle CSP:

- `tWALKcsp` that walk through the solutions of the CSP.
- `tCONST` that negates the constraint one-by-one and output 1 or more samples for each negate constraint.

constraints_highlight If set to `True`, the value of the nodes implied in a CSP (that could be specified through the keyword `constraint`) are highlighted in the console, given the Logger parameter `highlight_marked_nodes` is set to True.

3.2 Value Types

The current types usable within a terminal node are listed in this section. Each category (`Integer`, `String`, `BitField`) supports different parameters that allows to more accurately specify a data model, which enables `fuddly` to perform more enhanced fuzzing.

Note: These parameters will be especially leveraged by the generic disruptor `tTYPE` (`framework.generic_data_makers.d_fuzz_typed_nodes`). Refer to *Generic Disruptors* for more information on it, and to *Defining Specific Disruptors*, for how to create your own *disruptors*.

3.2.1 Integer

All integer types listed below provide the same interface (*framework.value_types.INT*). Their constructor take the following parameters:

values [optional, default value: None] List of the integers that are considered valid for the node backed by this *Integer object*. The default value is the first element of the list.

min [optional, default value: None] Minimum valid value for the node backed by this *Integer object*.

max [optional, default value: None] Maximum valid value for the node backed by this *Integer object*.

default [optional, default value: None] If not None, this value will be provided by default at first when *framework.value_types.INT.get_value()* is called.

determinist [default value: True] If set to `True` generated values will be in a deterministic order, otherwise in a random order.

This parameter is for internal usage and will always follow the *hosting* node instructions. If you want to change the deterministic order you have to do it at the node level by using the data model keyword `determinist` (refer to *Keywords to Describe Node Properties*).

values_desc [optional, default value: None] Dictionary that maps integer values to their descriptions (character strings). Leveraged for display purpose. Even if provided, all values do not need to be described.

All these parameters are optional. If you don't specify all of them the constructor will let more freedom within the data model. But if you have accurate information, don't hesitate to add them in the data model, as it does not weaken the test cases that will be generated by the generic disruptors, quite the opposite.

Below the different currently defined integer types, and the corresponding outputs for a data generated from them:

- *framework.value_types.UINT8*: unsigned integer on 8 bit
- *framework.value_types.SINT8*: signed integer on 8 bit (2's complement)
- *framework.value_types.UINT16_be*: unsigned integer on 16 bit, big endian
- *framework.value_types.UINT16_le*: unsigned integer on 16 bit, little endian
- *framework.value_types.SINT16_be*: signed integer on 16 bit (2's complement), big endian
- *framework.value_types.SINT16_le*: signed integer on 16 bit (2's complement), little endian
- *framework.value_types.UINT32_be*: unsigned integer on 32 bit, big endian
- *framework.value_types.UINT32_le*: unsigned integer on 32 bit, little endian
- *framework.value_types.SINT32_be*: signed integer on 32 bit (2's complement), big endian
- *framework.value_types.SINT32_le*: signed integer on 32 bit (2's complement), little endian
- *framework.value_types.UINT64_be*: unsigned integer on 64 bit, big endian
- *framework.value_types.UINT64_le*: unsigned integer on 64 bit, little endian
- *framework.value_types.SINT64_be*: signed integer on 64 bit (2's complement), big endian
- *framework.value_types.SINT64_le*: signed integer on 64 bit (2's complement), little endian
- *framework.value_types.INT_str*: ASCII encoded integer

For *framework.value_types.INT_str*, additional parameters are available:

base [optional, default value: 10] Numerical base that have to be used to represent the integer into a string

letter_case [optional, default value: 'upper'] Only for hexadecimal base. It could be `'upper'` or `'lower'` for representing hexadecimal numbers with these respective letter cases.

min_size [optional, default value: None] If specified, the integer representation will have a minimum size (with added zeros when necessary).

reverse [optional, default value: False] Reverse the order of the string if set to `True`.

3.2.2 String

All string types listed below provide the same interface (*framework.value_types.String*). Their constructor take the following parameters:

values [optional, default value: None] List of the character strings that are considered valid for the node backed by this *String object*. The default string is the first element of the list.

size [optional, default value: None] Valid character string size for the node backed by this *String object*.

min_sz [optional, default value: None] Minimum valid size for the character strings for the node backed by this *String object*. If not set, this parameter will be automatically inferred by looking at the parameter `values` whether this latter is provided.

max_sz [optional, default value: None] Maximum valid size for the character strings for the node backed by this *String object*. If not set, this parameter will be automatically inferred by looking at the parameter `values` whether this latter is provided.

deteterminist [default value: True] If set to `True` generated values will be in a deterministic order, otherwise in a random order.

This parameter is for internal usage and will always follow the *hosting* node instructions. If you want to change the deterministic order you have to do it at the node level by using the data model keyword `determinist` (refer to *Keywords to Describe Node Properties*).

codec [default value: 'latin-1'] Codec to use for encoding the string (e.g., 'latin-1', 'utf8'). Note that depending on the charset, additional fuzzing cases are defined.

case_sensitive [default value: True] If the string is set to be case sensitive then specific additional test cases will be generated in fuzzing mode.

default [optional, default value: None] If not None, this value will be provided by default at first when *framework.value_types.String.get_value()* is called.

extra_fuzzy_list [optional, default value: None] During data generation, if this parameter is specified with some specific values, they will be part of the test cases generated by the generic disruptor tTYPE.

absorb_regexp [optional, default value: None] You can specify a regular expression in this parameter as a supplementary constraint for data absorption operation (refer to *Absorption of Raw Data that Complies to the Data Model* for more information on that topic).

alphabet [optional, default value: string.printable] The alphabet to use for generating data, in case no `values` is provided. Also use during absorption to validate the contents. It is checked if there is no `values`.

values_desc [optional, default value: None] Dictionary that maps string values to their descriptions (character strings). Leveraged for display purpose. Even if provided, all values do not need to be described.

max_encoded_sz [optional, default value: None] Only relevant for subclasses that leverage the encoding infrastructure. Enable to provide the maximum legitimate size for an encoded string.

encoding_arg [optional, default value: None] Only relevant for subclasses that leverage the encoding infrastructure and that allow their encoding scheme to be configured. This parameter is directly provided to `framework.value_types.String.init_encoding_scheme()`.

Some String subclasses leverage the `String` encoding infrastructure, that enables to handle transparently any encoding scheme:

- The input values are the same as for the `String` type.
- Fuzzing test cases are generated based on the raw values, and then are encoded properly.
- Some test cases may be defined on the encoding scheme itself.

Note: To define a `String` subclass handling a specific encoding, you first have to define an encoder class that inherits from *framework.encoders.Encoder* (you may also use an existing one, if it fits your needs). Then you have to create a subclass of String decorated by *framework.value_types.from_encoder()* with your encoder class in parameter. Additionally, you can overload *framework.value_types.String.encoding_test_cases()* if you want to implement specific test cases related to your encoding. They will be automatically added to the set of test cases to be triggered by the disruptor `tTYPE`.

Note that the encoder you defined can also be used by a non-terminal node (refer to *How to Describe a Data Format With Some Encoded Parts*).

Below the different currently defined string types:

- *framework.value_types.String*: General purpose character string.
- *framework.value_types.Filename*: Filename. Similar to the type `String`, but some disruptors like `tTYPE` will generate more specific test cases.
- *framework.value_types.FolderPath*: FolderPath. Similar to the type `Filename`, but generated test cases are slightly different.
- *framework.value_types.GZIP*: `String` compressed with `zlib`. The parameter `encoding_arg` is used to specify the level of compression (0-9).
- *framework.value_types.GSM7bitPacking*: `String` encoded in conformity with `GSM 7-bits` packed format.
- *framework.value_types.Wrapper*: to be used as a mean to wrap a `String` with a prefix and/or a suffix, without defining specific *nodes* for that (meaning you don't need to model that part and want to simplify your data description).

3.2.3 BitField

The type *framework.value_types.BitField* takes the following parameters:

`subfield_limits` [optional, default value: None] List of the limits of each sub-fields (mutually exclusive with `subfield_sizes`), expressed in increasing order. For instance a limit list `[2, 6]` defines the sub-fields `0..1` (2 bits size) and `2..5` (4 bits size), for a total `BitField` size of 6 bits. Note that the list begin from the least significant sub-field to the more significant sub-field.

`subfield_sizes` [optional, default value: None] List of the size of each sub-fields (mutually exclusive with `subfield_limits`), beginning from the least significant sub-field to the more significant sub-field.

`subfield_values` [optional, default value: None] List of valid values for each sub-fields. Look at the following examples for usage. For each sub-field value list, the first value is the default.

`subfield_val_extremums` [optional, default value: None] List of minimum and maximum value for each sub-fields. Look at the following examples for usage.

`padding` [default value: 0] Should be either set to `0` or `1` for completion of the `Bitfield` to a byte boundary if it is not a byte-multiple. Note that the method *framework.value_types.BitField.extend_right()* allows to merge two `BitField` which could result in padding deletion.

lsb_padding [default value: True] If there is a need for padding, it will be added next to the least significant bit if this parameter is set to `True`, otherwise next to the most significant bit. This operation is performed *before* endianness encoding.

endian [default value: VT.LittleEndian] Endianness for *encoding* the BitField.

determinist [default value: True] If set to `True` generated values will be in a deterministic order, otherwise in a random order. Note that in *determinist mode*, all the values such a `BitField` should be able to generate are not covered but only a subset of them (i.e., all combinations are not computed). It has been chosen to only keep the value based on the following algorithm: "exhaust each subfield one at a time". The rationale is that in most cases, computing all combinations does not make sense, especially for fuzzing purpose. Additionally, note that such nominal generation are not the one used by the generic disruptor `tTYPE` which rely on `BitField` *fuzzy mode* (reachable through *framework.value_types.VT_Alt.enable_fuzz_mode()*).

This parameter is for internal usage and will always follow the *hosting* node instructions. If you want to change the deterministic order you have to do it at the node level by using the data model keyword `determinist` (refer to *Keywords to Describe Node Properties*).

default [optional, default value: None] If not None, it should be the list of default value for each sub-field. They will be provided by default at first when *framework.value_types.BitField.get_value()* is called.

subfield_descs [optional, default value: None] List of descriptions (character strings) for each sub-field. To describe only part of the sub-fields, put a `None` item for the others. This parameter is used for display purpose. Look at the following examples for usage.

subfield_value_descs [optional, default value: None] Dictionary providing descriptions (character strings) for values in each sub-field. More precisely, the dictionary maps subfield indexes to other dictionaries whose provides the mapping between values and descriptions. Leveraged for display purpose. Even if provided, all values do not need to be described. Look at the following examples for usage.

Let's take the following examples to make `BitField` usage obvious. On the first one, we specify the sub-fields of the `BitField` by their limit, and for each sub-field we give either a list of valid values, or a tuple expressing the minimum and maximum values. For the purpose of this example we use it directly, without going through the definition of a data model (for this topic refer to *Data Modeling* and *Defining the Imaginary MyDF Data Model*):

```
t = BitField(subfield_limits=[2,6,10,12],
             subfield_values=[[4,2,1], [2,15,16,3], None, [1]],
             subfield_val_extremums=[None, None, [3,11], None],
             padding=0, lsb_padding=True, endian=VT.LittleEndian)

t.pretty_print()

# output of the previous call:
#
#     (+|3: 01 |2: 0100 |1: 1111 |0: 10 |padding: 0000 |-) 19616
```

Note that the output is the first generated value from your description. To get another one you will have to call *framework.value_types.BitField.get_value()* on it. Obviously, this kind of stuff is done automatically for you during a fuzzing session.

On the second example we specify the sub-fields of the `BitField` by their sizes. And the other parameters are described in the same way as the first example. We additionally specify the parameter `subfield_descs` and `subfield_value_descs`. Look at the output for the differences.

```
t = BitField(subfield_sizes=[4,4,4],
             subfield_values=[[4,2,1], None, [10,13]],
             subfield_val_extremums=[None, [14, 15], None],
             padding=0, lsb_padding=False, endian=VT.BigEndian,
```

(continues on next page)

(continued from previous page)

```
            subfield_descs=['first', None, 'last'],
            subfield_value_descs={0:{4:'optionA',2:'optionB'}})

t.pretty_print()

# output of the previous call:
#
#     (+|padding: 0000 |2(last): 1101 |1: 1111 |0(first): 0100 [optionA] |-) 2788
```

See also:

Methods are defined to help for modifying a *framework.value_types.BitField*. If you want to deal with `BitField` in your specific disruptors, take a look especially at:

- *framework.value_types.BitField.set_subfield()*, *framework.value_types.BitField.get_subfield()*
- *framework.value_types.BitField.extend_right()*
- *framework.value_types.BitField.reset_state()*, *framework.value_types.BitField.rewind()*
- *framework.value_types.VT_Alt.enable_fuzz_mode()* (used currently by the disruptor `tTYPE`)

3.3 Helpers

3.3.1 Generator Node Templates

Hereunder are presented the currently available *generator-node* templates (which are defined in *framework.dmhelpers.generic*):

framework.dmhelpers.generic.LEN() Return a *generator* that returns the length of a node parameter.

framework.dmhelpers.generic.QTY() Return a *generator* that returns the quantity of child node instances (referenced by name) of the node parameter provided to the *generator*.

framework.dmhelpers.generic.TIMESTAMP() Return a *generator* that returns the current time (in a String node).

framework.dmhelpers.generic.CRC() Return a *generator* that returns the CRC (in the chosen type) of all the node parameters.

framework.dmhelpers.generic.WRAP() Return a *generator* that returns the result (in the chosen type) of the provided function applied on the concatenation of all the node parameters.

framework.dmhelpers.generic.CYCLE() Return a *generator* that iterates other the provided value list and returns at each step a node corresponding to the current value.

framework.dmhelpers.generic.OFFSET() Return a *generator* that computes the offset of a child node within its parent node.

framework.dmhelpers.generic.COPY_VALUE() Return a *generator* that retrieves the value of another node, and then return a *vt* node with this value.

framework.dmhelpers.generic.SELECT() Return a *generator* that select a subnode from a non-terminal node and return it

3.3.2 Block Builders

As well as *Generator Node Templates*, helpers of another kind are defined within the framework to make easier the modeling of some data formats. Basically, it is a bank of block builders that you can use to simplify the process of modeling if they match your needs.

These helpers are provided within `framework.dmhelpers`. The currently available helper modules are presented hereunder:

framework.dmhelpers.xml provides helpers for modeling XML tags (*framework.dmhelpers.xml.tag_builder()*). Note the helpers provide you with a precise data model which enables you to fuzz at XML level as well as at content level or to only focus on the content.

For example, the following call:

```
import framework.dmhelpers.xml as xml

xml_desc = \
xml.tag_builder('C1', params={'p1':'a', 'p2': ['foo', 'bar'], 'p3': 'c'},
                struct_mutable=False, tag_name_mutable=True, determinist=False,
                contents= \
                {'name': 'elt-content',
                 'contents': UINT16_be(values=[60,70,80])}, node_name='xml_sample')
```

will result in the following detailed data model:

```
xml_desc = \
{'name': 'xml_sample',
 'separator': {'contents': {'name': ('nl', uuid.uuid1()),
                            'contents': String(values=['\n'], max_sz=100,
                                               absorb_regexp='[\r\n|\n]+', codec=
↪'latin-1'),
                            'absorb_csts': AbsNoCsts(regexp=True)},
               'prefix': False, 'suffix': False, 'unique': False},
 'contents': [
     {'name': ('start-tag', uuid.uuid1()),
      'contents': [
          {'name': 'prefix',
           'contents': String(values=['<'], codec='latin-1'),
           'mutable': False, 'set_attrs': MH.Attr.Separator},

          {'name': ('content', uuid.uuid1()),
           'random': True,
           'separator': {'contents': {'name': ('spc', uuid.uuid1()),
                                      'contents': String(values=[' '], max_sz=100,
                                                         absorb_regexp='\s+',␣
↪codec='latin-1'),
                                      'mutable': False,
                                      'absorb_csts': AbsNoCsts(size=True,␣
↪regexp=True)},
                         'prefix': False, 'suffix': False, 'unique': False},
           'contents': [

               {'name': ('tag_name', uuid.uuid1()),
                'contents': String(values=['C1'], codec='latin-1'),
```

(continues on next page)

(continued from previous page)

```
                'mutable': True},

               {'section_type': MH.FullyRandom,
                'contents': [
                   {'name': ('attr1', uuid.uuid1()),
                    'contents': [
                        {'name': ('key', 1...), 'contents': String(values=['p1'], codec='latin-1')},
                        {'name': ('eq', 1...), 'contents': String(values=['='], codec='latin-1'),
                         'set_attrs': MH.Attr.Separator, 'mutable': False},
                        {'name': ('sep', 1...), 'contents': String(values=['"'], codec='latin-1'),
                         'set_attrs': MH.Attr.Separator, 'mutable': False},
                        {'name': ('val', 1...), 'contents': String(values=['a'], codec='latin-1')},
                        {'name': ('sep', 1...)},
                    ]},
                   {'name': ('attr2', uuid.uuid1()),
                    'contents': [
                        {'name': ('key', 2...), 'contents': String(values=['p2'], codec='latin-1')},
                        {'name': ('eq', 2...), 'contents': String(values=['='], codec='latin-1'),
                         'set_attrs': MH.Attr.Separator, 'mutable': False},
                        {'name': ('sep', 2...), 'contents': String(values=['"'], codec='latin-1'),
                         'set_attrs': MH.Attr.Separator, 'mutable': False},
                        {'name': ('val', 2...), 'contents': String(values=['foo', 'bar'], codec='latin-1')},
                        {'name': ('sep', 2...)},
                    ]},
                   {'name': ('attr3', uuid.uuid1()),
                    'contents': [
                        {'name': ('key', 3...), 'contents': String(values=['p3'], codec='latin-1')},
                        {'name': ('eq', 3...), 'contents': String(values=['='], codec='latin-1'),
                         'set_attrs': MH.Attr.Separator, 'mutable': False},
                        {'name': ('sep', 3...), 'contents': String(values=['"'], codec='latin-1'),
                         'set_attrs': MH.Attr.Separator, 'mutable': False},
                        {'name': ('val', 3...), 'contents': String(values=['c'], codec='latin-1')},
                        {'name': ('sep', 3...)},
                    ]}
                ]},

               {'name': ('suffix', uuid.uuid1()),
                'contents': String(values=['>'], codec='latin-1'),
                'mutable': False, 'set_attrs': MH.Attr.Separator}
```

(continues on next page)

(continued from previous page)

```
      ]},

     {'name': 'elt-content',
      'contents': UINT16_be(values=[60,70,80])},

     {'name': ('end-tag', uuid.uuid1()),
      'contents': [
         {'name': ('prefix', uuid.uuid1()),
          'contents': String(values=['</'], codec='latin-1'),
          'mutable': False, 'set_attrs': MH.Attr.Separator},
         {'name': ('content', uuid.uuid1()),
          'contents': String(values=['C1'], codec='latin-1'),
          'mutable': True},
         {'name': ('suffix', uuid.uuid1()),
          'contents': String(values=['>'], codec='latin-1'),
          'mutable': False, 'set_attrs': MH.Attr.Separator},
      ]}
 ]}
```

3.4 Data Model Patterns

3.4.1 How to Describe Different Shapes for Some Parts of Data

To describe different forms for a non-terminal node, you can define it in terms of shapes like illustrated by the example below:

```
    {'name': 'shape',
     'separator': {'contents': {'name': 'sep',
                                'contents': String(values=[' [!] '])}},
     'contents': [

         ### SHAPE 1 ####
         {'weight': 20,
          'contents': [
              {'name': 'prefix1',
               'contents': String(size=10, alphabet='+')},

              {'name': 'body_top',
               'contents': [

                   {'name': 'body',
                    'separator': {'contents': {'name': 'sep2',
                                               'contents': String(values=['::'])}},
                    'shape_type': MH.Random,
                    'contents': [
                        {'contents': String(values=['AAA']),
                         'qty': (0, 4),
                         'name': 'str1'},
                        {'contents': String(values=['42']),
                         'name': 'str2'}
```

(continues on next page)

(continued from previous page)

```
                    ]}
                ]}

            ]},

            ### SHAPE 2 ###
            {'weight': 20,
             'contents': [
                 {'name': 'prefix2',
                  'contents': String(size=10, alphabet='>')},

                 {'name': 'body'}
             ]}
        ]}
```

The shapes are ordered by their weight. In *deterministic* mode (refer to *Data Model Keywords*) that means a non terminal-node will be sequentially resolved from its heavier shape to its lighter shape. In *random* mode, the weight are used in a probabilistic way.

The example above also illustrates how to represent an *optional part* in the description of a data format (within the first shape of the example, line 20-22). You only have to set the minimum quantity of a node to `0` (line 21), and it will be considered as an optional part.

If you iterate over this data model with `tWALK(nt_ony=True)` (refer to *Generic Disruptors*) you will see the various data forms understood by `fuddly` which would be leveraged by most of the generic stateful disruptors.

```
# First Form
[!] ++++++++++ [!] ::42:: [!]

# Second Form
[!] ++++++++++ [!] ::AAA::AAA::42:: [!]

# Third Form
[!] >>>>>>>>>> [!] ::AAA::AAA::42:: [!]
```

As you can see, the first and second forms are from `SHAPE 1`. The differences between them comes from the optional part: the first form does not have the optional part while the second one includes it. Finally, the third form is from the `SHAPE 2`.

See also:

Refer to *The Model Walker Infrastructure* for more information on the *Model Walker* infrastructure which makes really easy the implementation of stateful disruptors leveraging the different forms of a data.

See also:

Refer to *How to Describe a Data Format Whose Parts Change Depending on Some Fields* if you need to change the data format depending on the existence of optional parts.

3.4.2 How to Describe the Separators of a Data Format

The example below shows how to define the separators for delimiting lines of an imaginary data model (line 2-7), and for delimiting parameters with space characters (line 12-14).

```
1  {'name': 'separator_test',
2   'separator': {'contents': {'name': 'sep',
3                              'contents': String(values=['\n'], absorb_regexp='[\r\n|\n]+
   ↪'),
4                              'absorb_csts': AbsNoCsts(regexp=True)},
5                 'prefix': False,
6                 'suffix': False,
7                 'unique': True},
8   'contents': [
9       {'section_type': MH.FullyRandom,
10       'contents': [
11           {'name': 'parameters',
12            'separator': {'contents': {'name': ('sep',2),
13                                       'contents': String(values=[' '], absorb_regexp=' +
   ↪'),
14                                       'absorb_csts': AbsNoCsts(regexp=True)}},
15            'qty': 3,
16            'contents': [
17                {'section_type': MH.FullyRandom,
18                 'contents': [
19                     {'name': 'color',
20                     'contents': [
21                         {'name': 'id',
22                          'contents': String(values=['color='])},
23                         {'name': 'val',
24                          'contents': String(values=['red', 'black'])}
25                     ]},
26                     {'name': 'type',
27                      'contents': [
28                          {'name': ('id', 2),
29                           'contents': String(values=['type='])},
30                          {'name': ('val', 2),
31                           'contents': String(values=['circle', 'cube', 'rectangle'],␣
   ↪determinist=False)}
32                     ]},
33                ]}]},
34           {'contents': String(values=['AAAA', 'BBBB', 'CCCC'], determinist=False),
35            'qty': (4, 6),
36            'name': 'str'}
37       ]}
38   ]}
```

From this data model you could get a data like that:

```
CCCC
BBBB
 type=circle color=red
 type=rectangle color=red
BBBB
```

(continues on next page)

(continued from previous page)

```
AAAA
CCCC
 color=red type=cube
```

Note: Note this data model can be used to absorb data samples (refer to *Absorption of Raw Data that Complies to the Data Model*) that may use more than one empty line as first-level separator (thanks to the `absorb_regexp` parameter in line 3), and more than one space character as second-level separators (thanks to the `absorb_regexp` parameter in line 13).

Note: You can also perform specific *separator mutation* within a disruptor (refer to *Defining Specific Disruptors*), as separator nodes have the specific attribute *framework.node.NodeInternals.Separator* set.

3.4.3 How to Describe a Data Format Whose Parts Change Depending on Some Fields

The example below shows how to define a data format based on *opcodes* and *sub-opcodes* which change the form of the data itself. We use for that purpose the keyword `exists_if` with some subclasses of *framework.node.NodeCondition* and node references.

Note: The keyword `exists_if` can directly take a node reference. In such case, the condition is the existence of this node itself.

```
1  {'name': 'exist_cond',
2   'shape_type': MH.Ordered,
3   'contents': [
4       {'name': 'opcode',
5        'contents': String(values=['A1', 'A2', 'A3'], determinist=True)},
6
7       {'name': 'command_A1',
8        'contents': String(values=['AAA', 'BBBB', 'CCCCC']),
9        'exists_if': (RawCondition('A1'), 'opcode'),
10       'qty': 3},
11
12      {'name': 'command_A2',
13       'contents': UINT32_be(values=[0xDEAD, 0xBEEF]),
14       'exists_if': (RawCondition('A2'), 'opcode')},
15
16      {'name': 'command_A3',
17       'exists_if': (RawCondition('A3'), 'opcode'),
18       'contents': [
19           {'name': 'A3_subopcode',
20            'contents': BitField(subfield_sizes=[15,2,4], endian=VT.BigEndian,
21                                 subfield_values=[None, [1,2], [5,6,12]],
22                                 subfield_val_extremums=[[500, 600], None, None],
23                                 determinist=False)},
24
```

(continues on next page)

(continued from previous page)

```
            {'name': 'A3_int',
             'contents': UINT16_be(values=[10, 20, 30], determinist=False)},

            {'name': 'A3_deco1',
             'exists_if': (IntCondition(10), 'A3_int'),
             'contents': String(values=['*1*0*'])},

            {'name': 'A3_deco2',
             'exists_if': (IntCondition([20, 30]), 'A3_int'),
             'contents': String(values=['+2+0+3+0+'])}
        ]},

       {'name': 'A31_payload',
        'contents': String(values=['$ A31_OK $', '$ A31_KO $'], determinist=False),
        'exists_if': (BitFieldCondition(sf=2, val=[6,12]), 'A3_subopcode')},

       {'name': 'A32_payload',
        'contents': String(values=['$ A32_VALID $', '$ A32_INVALID $'],
↪determinist=False),
        'exists_if': (BitFieldCondition(sf=[0, 1, 2], val=[[500, 501], [1, 2], 5]), 'A3_
↪subopcode')}
    ]}
```

Note: Existence condition does not have to be located after the node you want to check, it can also be located before. Fuddly will postpone the condition checking in this case.

Example of data generated by such a data model are presented below (in ASCII art):

```
[0] exist_cond [NonTerm]
 \__(1) exist_cond/opcode [String] size=2B
 |        \_raw: 'A3'
 \__[1] exist_cond/command_A3 [NonTerm]
 |   \__(2) exist_cond/command_A3/A3_subopcode [BitField] size=3B
 |   |        \_ (+|2: 0110 |1: 01 |0: 0000010010010001 |padding: 000 |-) 6558280
 |   |        \_raw: 'd\x12H'
 |   \__(2) exist_cond/command_A3/A3_int [UINT16_be] size=2B
 |   |        \_ 10 (0xA)
 |   |        \_raw: '\x00\n'
 |   \__(2) exist_cond/command_A3/A3_deco1 [String] size=5B
 |            \_raw: '*1*0*'
 \__(1) exist_cond/A31_payload [String] size=10B
          \_raw: '$ A31_OK $'


[0] exist_cond [NonTerm]
 \__(1) exist_cond/opcode [String] size=2B
 |        \_raw: 'A1'
 \__(1) exist_cond/command_A1 [String] size=3B
 |        \_raw: 'AAA'
 \__(1) exist_cond/command_A1:2 [String] size=3B
 |        \_raw: 'AAA'
```

(continues on next page)

(continued from previous page)

```
 \__(1) exist_cond/command_A1:3 [String] size=3B
          \_raw: 'AAA'


[0] exist_cond [NonTerm]
 \__(1) exist_cond/opcode [String] size=2B
 |        \_raw: 'A2'
 \__(1) exist_cond/command_A2 [UINT32_be] size=4B
          \_ 48879 (0xBEEF)
          \_raw: '\x00\x00\xbe\xef'
```

Note: Note this data model can be used for generating data and also (without modification) for absorbing data samples that comply to its grammar (refer to *Absorption of Raw Data that Complies to the Data Model*)

3.4.4 How to Generate Nodes Dynamically (for length, counter, ...)

The example below shows how to describe a node that will dynamically generate a node containing the length of another one, a variable character string in our case.

```
{'name': 'len_gen',
 'contents': [
     {'name': 'len',
      'contents': lambda x: Node('cts', value_type= \
                                 UINT32_be(values=[len(x.to_bytes())])),
      'node_args': 'payload'},

     {'name': 'payload',
      'contents': String(min_sz=10, max_sz=100, determinist=False)},
 ]}
```

Note the *generator* is just a specific kind of node (*framework.node.NodeInternals_GenFunc*) that embeds a function that returns a node (*framework.node.Node*). In the previous description, the function is provided through the keyword `contents`, and it's a simple lambda function taking a node as parameter, on which is called *framework.node.Node.to_bytes()* to get its bytes representation and then the `len()` function. The result is used for defining a terminal node of type *framework.value_types.UINT32_be* (refer to section *Integer*).

This use case can be described by using the specific *generator template* *framework.dmhelpers.generic.LEN()* which will basically return the previous lambda function. The following example makes use of it.

Note: Generator templates are defined as static methods of *framework.dmhelpers.generic.MH*. They make the description of some generic use cases simpler.

```
{'name': 'len_gen',
 'contents': [
     {'name': 'len',
      'contents': LEN(UINT32_be),
      'node_args': 'payload'},

```

(continues on next page)

(continued from previous page)

```
    {'name': 'payload',
     'contents': String(min_sz=10, max_sz=100, determinist=False)},
 ]}
```

To conclude on this use case, note that the previous description can be used for data generation, but it won't be usable as-is for data absorption (refer to *Absorption of Raw Data that Complies to the Data Model*). Indeed, the way absorption works is by walking through the graph and it will reach the generator first. This one will freeze the string contents by getting its bytes representation and will create an `UINT32_be` node with only one value, the length of the arbitrarily generated string. This value will be used for validating the corresponding data part within the raw data to absorb, as the absorption operation will by default enforce contents equality. Hence, it will fail. To solve this problem, the simplest solution is to release some local constraints during absorption, namely we need to release the `Contents` constraint for the `len` node. More simply, we can release all the absorption constraints for this node, as shown in the following example:

```
{'name': 'len_gen',
 'contents': [
    {'name': 'len',
     'contents': LEN(UINT32_be),
     'node_args': 'payload',
     'absorb_csts': AbsNoCsts()  # or more accurately AbsCsts(contents=False)
     },

    {'name': 'payload',
     'contents': String(min_sz=10, max_sz=100, determinist=False)},
 ]}
```

Another solution can be to define an alternate configuration that will be used only for absorption:

```
{'name': 'len_gen',
 'contents': [
    {'name': 'len',
     'contents': LEN(UINT32_be),
     'node_args': 'payload',
     'alt': [
         {'conf': 'ABS',
          'contents': UINT32_be(max=100)} ]},

    {'name': 'payload',
     'contents': String(min_sz=10, max_sz=100, determinist=False)},
 ]}
```

This solution is more complex, but can revealed itself to be useful for more complex situation.

See also:

Look at the example *ZIP archive modification* to see how to change the node configuration before absorption. And for more insights on that topic refer to *Data Modeling* and *Defining Specific Disruptors*.

Finally, let's take the following example that illustrates other *generator templates*, namely *framework.dmhelpers.generic.QTY()*, *framework.dmhelpers.generic.CRC()* and *framework.dmhelpers.generic.TIMESTAMP()*.

```
{'name': 'misc_gen',
 'contents': [
```

(continues on next page)

(continued from previous page)

```
     {'name': 'integers',
      'contents': [
          {'name': 'int16',
           'qty': (2, 10),
           'contents': UINT16_be(values=[16, 1, 6], determinist=False)},

          {'name': 'int32',
           'qty': (3, 8),
           'contents': UINT32_be(values=[32, 3, 2], determinist=False)}
      ]},

     {'name': 'int16_qty',
      'contents': QTY(node_name='int16', vt=UINT8),
      'node_args': 'integers'},

     {'name': 'int32_qty',
      'contents': QTY(node_name='int32', vt=UINT8),
      'node_args': 'integers'},

     {'name': 'tstamp',
      'contents': TIMESTAMP("%H%M%S"),
      'absorb_csts': AbsCsts(contents=False)},

     {'name': 'crc',
      'contents': CRC(UINT32_be),
      'node_args': ['tstamp', 'int32_qty'],
      'absorb_csts': AbsCsts(contents=False)}
 ]}
```

Note: Note this data model is compatible for *data absorption*.

Here under an example of data generated by such a data model (in ASCII art):

```
[0] misc_gen [NonTerm]
 \__[1] misc_gen/integers [NonTerm]
 |   \__(2) misc_gen/integers/int16 [UINT16_be] size=2B
 |   |       \_ 6 (0x6)
 |   |       \_raw: '\x00\x06'
 |   \__(2) misc_gen/integers/int16:2 [UINT16_be] size=2B
 |   |       \_ 1 (0x1)
 |   |       \_raw: '\x00\x01'
 |   \__(2) misc_gen/integers/int16:3 [UINT16_be] size=2B
 |   |       \_ 1 (0x1)
 |   |       \_raw: '\x00\x01'
 |   \__(2) misc_gen/integers/int16:4 [UINT16_be] size=2B
 |   |       \_ 6 (0x6)
 |   |       \_raw: '\x00\x06'
 |   \__(2) misc_gen/integers/int16:5 [UINT16_be] size=2B
 |   |       \_ 6 (0x6)
 |   |       \_raw: '\x00\x06'
 |   \__(2) misc_gen/integers/int16:6 [UINT16_be] size=2B
```

(continues on next page)

(continued from previous page)

```
|   |        \_ 1 (0x1)
|   |        \_raw: '\x00\x01'
|   \__(2) misc_gen/integers/int16:7 [UINT16_be] size=2B
|   |        \_ 1 (0x1)
|   |        \_raw: '\x00\x01'
|   \__(2) misc_gen/integers/int32 [UINT32_be] size=4B
|   |        \_ 2 (0x2)
|   |        \_raw: '\x00\x00\x00\x02'
|   \__(2) misc_gen/integers/int32:2 [UINT32_be] size=4B
|   |        \_ 3 (0x3)
|   |        \_raw: '\x00\x00\x00\x03'
|   \__(2) misc_gen/integers/int32:3 [UINT32_be] size=4B
|            \_ 2 (0x2)
|            \_raw: '\x00\x00\x00\x02'
\__[1] misc_gen/int16_qty [GenFunc | node_args: misc_gen/integers]
|   \__(2) misc_gen/int16_qty/cts [UINT8] size=1B
|            \_ 7 (0x7)
|            \_raw: '\x07'
\__[1] misc_gen/int32_qty [GenFunc | node_args: misc_gen/integers]
|   \__(2) misc_gen/int32_qty/cts [UINT8] size=1B
|            \_ 3 (0x3)
|            \_raw: '\x03'
\__[1] misc_gen/tstamp [GenFunc | node_args: None]
|   \__(2) misc_gen/tstamp/cts [String] size=6B
|            \_raw: '170140'
\__[1] misc_gen/crc [GenFunc | node_args: misc_gen/tstamp, misc_gen/int32_qty]
    \__(2) misc_gen/crc/cts [UINT32_be] size=4B
             \_ 110906314 (0x69C4BCA)
             \_raw: '\x06\x9cK\xca'
```

Which correspond to the following data:

```
'\x00\x06\x00\x01\x00\x01\x00\x06\x00\x06\x00\x01\x00\x01\x00\x00\x00\x02\x00\x00\x00\
↪x03\x00\x00\x00\x02\x07\x03170140\x06\x9cK\xca'
```

See also:

You may delay the triggering of a generator, until everything else has been resolved. It is especially useful when you describe a generator that use a node with an existence condition and when this condition cannot be resolved at the time the generator will normally be triggered (that is when it is reached during the nominal graph traversal). To postpone this triggering, you have to set the generator-specific keyword `trigger_last` to *True*. Refer to *Data Model Keywords* for more information on the available keywords.

3.4.5 How to Describe a Data Format With Some Encoded Parts

The example below shows how to describe a data format with some parts encoded in different ways.

The non-terminal node named `enc` (lines 9-19) has the attribute `encoder` (refer to *Data Model Keywords*) which means that it will be encoded following the scheme of the specified encoder. In this case it is the *framework.encoders.GZIP_Enc* with a level of compression of 6. Within this node is also defined a typed node (lines 17-18) named `data1` which is encoded in *UTF16 little endian* through the parameter `codec` of *framework.value_types.String*.

Note also the parameter `after_encoding=False` (lines 6 and 14), which is supported by every relevant generator node templates (refer to *Generator Node Templates*) and enable them to act either on the encoded form or the decoded form of their node parameters.

```
1  {'name': 'enc',
2   'contents': [
3       {'name': 'data0',
4        'contents': String(values=['Plip', 'Plop']) },
5       {'name': 'crc',
6        'contents': CRC(vt=UINT32_be, after_encoding=False),
7        'node_args': ['enc_data', 'data2'],
8        'absorb_csts': AbsFullCsts(contents=False) },
9       {'name': 'enc_data',
10       'encoder': GZIP_Enc(6),
11       'set_attrs': [NodeInternals.Abs_Postpone],
12       'contents': [
13          {'name': 'len',
14           'contents': LEN(vt=UINT8, after_encoding=False),
15           'node_args': 'data1',
16           'absorb_csts': AbsFullCsts(contents=False)},
17          {'name': 'data1',
18           'contents': String(values=['Test!', 'Hello World!'], codec='utf-16-le') },
19       ]},
20      {'name': 'data2',
21       'contents': String(values=['Red', 'Green', 'Blue']) }
22   ]}
```

This data description will enable you to produce data compliant to the specified encoding schemes in a transparent way. Additionally, any fuzzing operations (*Defining Specific Disruptors*) you want to perform on any data parts will be done *before* any encoding takes place.

If you want to perform some fuzzing on the encoding scheme itself you will have first to describe its format. Then it boils down to run some generic disruptors on them or some of your own. However, note that some value types that support encoding (refer to *Value Types*) embed specific test cases on the encoding scheme (which is the case for `utf-16-le`-encoded strings for instance).

Finally, absorption (refer to *Absorption of Raw Data that Complies to the Data Model*) is also supported when encoding is used within your data description. For instance, the following data will be absorbed by the previous data model:

```
b'Plop\x8c\xd6/\x06x\x9cc\raHe(f(aPd\x00\x00\x0bv\x01\xc7Blue'
```

To perform that operation you can write the following python code:

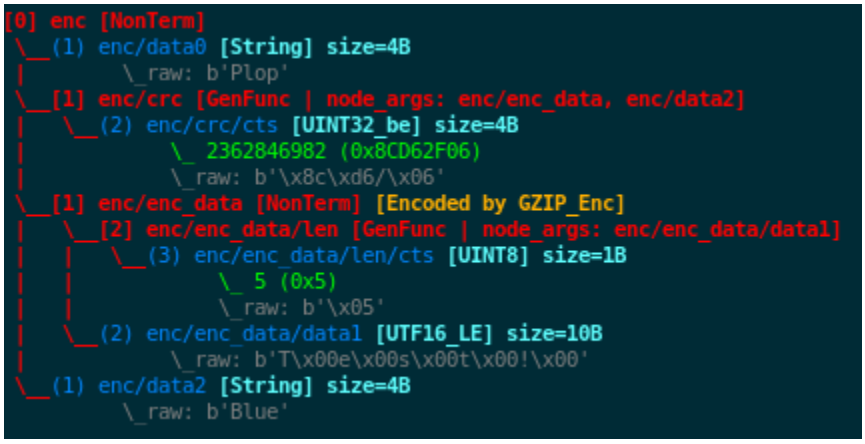
```
1  from framework.plumbing import *
2  from framework.node import AbsorbStatus
3
4  raw_data = b'Plop\x8c\xd6/\x06x\x9cc\raHe(f(aPd\x00\x00\x0bv\x01\xc7Blue'
```

(continues on next page)

(continued from previous page)

```
5
6  fmk = FmkPlumbing()
7  fmk.run_project(name="tuto")
8  enc_dm = fmk.dm.get_atom('enc')
9
10 status, off, size, name = enc_dm.absorb(raw_data, constraints=AbsFullCsts())
11 if status == AbsorbStatus.FullyAbsorbed:
12     enc_dm.show()
```

The following picture displays the result of the previous code (triggered by line 12):



Note: The `content` absorption constraint is released for the generator nodes `crc` (line 8) and `len` (line 16) in order to allow any value to be absorbed and not limit them to the value generated the last time the generators triggered (which occurs during node freezing). Indeed, generators based on these templates will dynamically generate a typed node that contains only one value—based on the current value their node parameters have while the generator is triggered.

Note: Line 11 is to make the absorption operation work correctly. Indeed because of the encoding, constraints are not rigid enough to make fuddly work out the absorption without some help.

3.4.6 How to Describe a Data Format That Contains Complex Strings

Parts of the data that only contain strings can easily be described using python's regular expressions. Here are some rules to respect:

- Using square brackets `[ ]` to indicate a set of characters will result in the creation of a *framework.value_types.String* terminal node that contains an *alphabet*. Likewise, the usage of `.` or meta-sequences such as `\s`, `\S`, `\w`, `\W`, `\d` or `\D` will lead to the creation of such type of nodes.
- Anything else will be translated into a *framework.value_types.String* terminal node that declares a list of values. `( )` can be used to delimit a portion of the regular expression that need to be translated into a terminal node on its own.

Note: If each item in a list of values are integers an *framework.value_types.INT_str* will be created instead of a *framework.value_types.String*.

- (,), [,], ?, *, +, {, }, |, \, -, . are the only recognised special characters. They cannot be used in an unsuitable context without being escaped (exceptions are made for |, . and -).
- Are only allowed regular expressions that can be translated into one terminal node or into one non-terminal node composed of terminal ones. If this rule is not respected an `framework.error_handling.InconvertibilityError` will be raised.
- An inconsistency between the charset and the characters that compose the regular expression will result in an `framework.error_handling.CharsetError`.

Note: The default charset used by Fuddly is `MH.Charset.ASCII_EXT`. To change this behaviour, use the keyword `charset` (refer to *Keywords to Describe Node Properties*).

To embody these rules, let's take some examples:

Example 1: The basics.

```
regex = {'name': 'HTTP_version',
         'contents': '(HTTP)/[0-9]\.(0|1|2|\x33|4|5|6|7|8|9)'}
# is equivalent to
classic = {'name': 'HTTP_version',
           'contents': [
              {'name': 'HTTP_version_1', 'contents': String(values=["HTTP"])},
              {'name': 'HTTP_version_2', 'contents': String(values=["/"])},
              {'name': 'HTTP_version_3',
               'contents': String(alphabet="0123456789", size=1)},
              {'name': 'HTTP_version_4', 'contents': String(values=["."])},
              {'name': 'HTTP_version_5', 'contents': INT_str(min=0, max=9)} ]}
```

Example 2: Introducing choices. (Refer to *Keywords to Describe Non Terminal Node*)

```
regex = {'name': 'something',
         'contents': '(333|444)|(foo|bar)|[\d]|[th|is]'}
# is equivalent to
classic = {'name': 'something',
           'shape_type': MH.Pick,
           'contents': [
              {'name':'something_1', 'contents':INT_str(values=[333, 444])},
              {'name':'something_2', 'contents':String(values=["foo", "bar"])},
              {'name':'something_3', 'contents':String(alphabet="0123456789",size=1)},
              {'name':'something_4', 'contents':String(alphabet="th|is", size=1)}
           ]}
```

Example 3: Using shapes. (Refer to *Data Model Patterns*)

```
regex = {'name': 'something',
         'contents': 'this[\d](is)|a|digit[!]'}
# is equivalent to
classic = {'name': 'something',
           'contents': [
              {'weight': 1,
               'contents': [
                  {'name': 'something_1', 'contents': String(values=['this'])},
                  {'name': 'something_2', 'contents': String(alphabet='0123456789')},
```

(continues on next page)

(continued from previous page)

```
                {'name': 'something_3', 'contents': String(values=['is'])},
             ]},

            {'weight': 1,
             'contents': [
                {'name': 'something_4', 'contents': String(values=['a'])},
             ]},

            {'weight': 1,
             'contents': [
                {'name': 'something_5', 'contents': String(values=['digit'])},
                {'name': 'something_6', 'contents': String(alphabet='!')},
             ]},
        ]}
```

Example 4: Using quantifiers and the escape character \.

```
regex = {'name': 'something',
         'contents': '\(this[is]{3,4}the+end\]'}
# is equivalent to
classic = {'name': 'something',
           'contents': [
              {'name': 'something_1', 'contents': String(values=["(this"])},
              {'name': 'something_2',
               'contents': String(alphabet="is", min_sz=3, max_sz=4)},
              {'name': 'something_3', 'contents': String(values=["th"])},
              {'name': 'something_4', 'qty': (1, -1),
               'contents': String(values=["e"])},
              {'name': 'something_5', 'contents': String(values=["end]"])} ]}
```

Example 5: Invalid regular expressions.

```
error_1 = {'name': 'rejected', 'contents': '(HT(T)P)/'}
# raise an framework.error_handling.InconvertibilityError
# because there are two nested parenthesis.

error_2 = {'name': 'rejected', 'contents': '(HT?TP)foo|bar'}
# raise also an framework.error_handling.InconvertibilityError
# because a quantifier (that requires the creation of a terminal node)
# has been found within parenthesis.
```

3.4.7 How to Describe Constraints of Data Formats

When some relations exist between various parts of the data format you want to describe you have different possibilities within `fuddly`:

- either using some specific keywords that capture basic constraints (e.g., `qty_from`, `sync_size_with`, `exists_if`, ...);
- or through Generator nodes (refer to *Generator Node Templates*);
- or by specifying a CSP through the keyword `constraint`, which leverage a constraint programming backend (currently limited to the `python-constraint` module)

The CSP specification case is described in more details in what follows. To describe constraints in the form of a CSP, you should use the `constraints` keyword that allows you to provide a list of *framework.constraint_helpers.Constraint* objects, which are the building blocks for specifying constraints between multiple nodes.

For instance, let's analyse the following data description (extracted from the `mydf` data model in `tuto.py`).

```
1     csp_desc = \
2         {'name': 'csp',
3          'constraints': [Constraint(relation=lambda d1, d2: d1[1]+1 == d2[0] or d1[1]+2 == d2[0],
4                                     vars=('delim_1', 'delim_2')),
5                         Constraint(relation=lambda x, y, z: x == 3*y + z,
6                                     vars=('x_val', 'y_val', 'z_val'))],
7          'constraints_highlight': True,
8          'contents': [
9              {'name': 'equation',
10              'contents': String(values=['x = 3y + z'])},
11             {'name': 'delim_1', 'contents': String(values=[' [', ' ('])},
12             {'name': 'variables',
13              'separator': {'contents': {'name': 'sep', 'contents': String(values=[','])},
14                            'prefix': False, 'suffix': False},
15              'contents': [
16                  {'name': 'x',
17                   'contents': [
18                       {'name': 'x_symbol',
19                        'contents': String(values=['x:', 'X:'])},
20                       {'name': 'x_val',
21                        'contents': INT_str(min=120, max=130)} ]},
22
23     [...]
```

You can see that two constraints have been specified (l.3-6) through the specific *framework.constraint_helpers.Constraint* objects. The constructor take a mandatory `relation` parameter expecting a boolean function that should express a relation between any nodes reachable from the non-terminal node on which the `constraints` keyword is attached. It takes also a `vars` parameter expecting a list of the names of the nodes used in the boolean function (in the same order as the parameters of the function).

Note: The `constraints` keyword can be used several times along the description, but all the specified *framework.constraint_helpers.Constraint* will eventually end up in a single CSP.

These constraints, will then be resolved at *framework.node.Node.freeze()* time (depending if the parameter `resolve_csp` is set to True). Note also that before resolving the CSP it is possible to fix the value of some variables by freezing the related nodes with the parameter `restrict_csp`. This is what is performed by the *framework.fuzzing_primitives.ModelWalker* infrastructure when walking a specific node which is part of a CSP, so that the walked node won't be modified further to the CSP solving process.

Note: The constructor of *framework.constraint_helpers.Constraint* takes also an optional parameter `var_to_varns` in order to support namespaces (used to discriminate nodes having identical name in the data description). Refer to `namespace` keyword for more details, and to the `csp_ns` node description in the data model `mydf` (in `tuto.py`).

CHAPTER

FOUR

DATA MANIPULATION

The following section will provide you with an understanding on how to manipulate modeled data with `fuddly` primitives. Data manipulation is what *disruptors* perform (refer to *Defining Specific Disruptors*). This chapter will enable you to write your own *disruptors* or to simply perform custom manipulation of a data coming from a file, retrieved from the network (thanks to data absorption—refer to *Absorption of Raw Data that Complies to the Data Model*), or even generated from scratch.

4.1 Overview

To guide you over what is possible to perform, let's consider the following data model:

```
from framework.node import *
from framework.value_types import *
from framework.node_builder import *

 example_desc = \
 {'name': 'ex',
  'contents': [
      {'name': 'data0',
       'contents': String(values=['Plip', 'Plop']) },

      {'name': 'data_group',
       'contents': [

          {'name': 'len',
           'mutable': False,
           'contents': LEN(vt=UINT8, after_encoding=False),
           'node_args': 'data1',
           'absorb_csts': AbsFullCsts(contents=False)},

          {'name': 'data1',
           'contents': String(values=['Test!', 'Hello World!']) },

          {'name': 'data2',
           'qty': (1,3),
           'semantics': ['sem1', 'sem2'],
           'contents': UINT16_be(min=10, max=0xa0ff),
           'alt': [
                {'conf': 'alt1',
                 'contents': SINT8(values=[1,4,8])},
```

(continues on next page)

(continued from previous page)

```
                {'conf': 'alt2',
                 'contents': UINT16_be(min=0xeeee, max=0xff56)} ]},

          {'name': 'data3',
           'semantics': ['sem2'],
           'sync_qty_with': 'data2',
           'contents': UINT8(values=[30,40,50]),
           'alt': [
                {'conf': 'alt1',
                 'contents': SINT8(values=[1,4,8])}]},
         ]},

      {'name': 'data4',
       'contents': String(values=['Red', 'Green', 'Blue']) }
  ]}
```

This is what we call a data descriptor. It cannot be used directly, it should first be transformed to `fuddly` internal representation based on *framework.node.Node*. The code below shows how to perform that:

```
nb = NodeBuilder()
rnode = nb.create_graph_from_desc(example_desc)
rnode.set_env(Env())
```

`fuddly` models data as directed acyclic graphs whose terminal nodes describe the different parts of a data format (refer to *Data Modeling*). In order to enable elaborated manipulations it also creates a specific object to share between all the nodes some common information related to the graph: the *framework.node.Env* object. You should note that we create this *environment* object and setup the root node with it. Actually it provides all the nodes of the graph with this environment. From now on it is possible to access the environment from any node, and `fuddly` is now able to deal with this graph.

Note: The method *framework.node_builder.NodeBuilder.create_graph_from_desc()* return a *framework.node.Node* which is the root of the graph.

Note: When you instantiate a modeled data from a model through *framework.data_model.DataModel.get_atom()* as illustrated in *Using fuddly Through Advanced Python Interpreter*, the environment object is created for you. Likewise, when you register a data descriptor through *framework.data_model.DataModel.register()* (refer to *Defining the Imaginary MyDF Data Model*), no need to worry about the environment.

Note: The *framework.node_builder.NodeBuilder* object which is used to create a graph from a data descriptor is bound to the graph and should not be used for creating another graph. It contains some information on the created graph such as a dictionary of all its nodes `mb.node_dico`.

4.1.1 Generate Data a.k.a. Freeze a Graph

If you want to get a data from the graph you have to freeze it first as it represents many different potential data at once (actually it acts like a template). To do so, just call the method *framework.node.Node.freeze()* on the root node. It will provide you with a nested set of lists containing the frozen value for each node selected within the graph to provide you with a data.

What is way more interesting in the general case is obtaining a byte string of the data. For this you just have to call *framework.node.Node.to_bytes()* on the root node which will first freeze the graph and then flatten the nested list automatically to provide you with the byte string.

If you want to get another data from the graph you should first unfreeze it because otherwise any further call to the previous methods will give you the same value. To do that you can call the method *framework.node.Node.unfreeze()*. You will then be able to get a new data by freezing it again. Actually doing so will produce the next data by cycling over the possible node values (described in the graph) in a random or a determinist way (refer to *Operations on Node Properties and Attributes*). If you look at getting data from the graph by walking over each of its nodes independently then you should look for instance at the generic disruptor `tWALK` (refer to *Generic Disruptors*) and also to the model walker infrastructure *The Model Walker Infrastructure*).

By default, `unfreeze` will act recursively and will affect every nodes reachable from the calling one. You can unfreeze only the node on which the method is called by switching its `recursive` parameter.

You may want to unfreeze the graph without changing its state, just because you performed some modifications locally and want it to be taken into account when getting a new data from the graph. (refer to *Node Configurations* for a usage example). For that purpose, you may use the `dont_change_state` parameter of *framework.node.Node.unfreeze()* which allows to unfreeze without cycling.

Another option you may want is to unfreeze only the constraints of your graph which based on existence conditions (refer to *How to Describe a Data Format Whose Parts Change Depending on Some Fields*), *generator* and *func* nodes. To do so, set the `reevaluate_constraints` parameter to `True`.

To cycle over the possible node values or shapes (for non terminal nodes) a state is kept. This state is normally reset automatically when the node is exhausted in order to cycle again. can be reset thanks to the method *framework.node.Node.reset_state()*. In addition to resetting the node state it also unfreezes it.

Note: When a cycle over the possible node values or shapes is terminated, a notification is raised (through the linked environment object). Depending on the `Finite` node attribute generic disruptors will recycle the node or change to another one. Setting the `Finite` property on all the graph will enable you to have an end on data generation, and to avoid the generation of duplicated data.

Finally if you want to unfreeze all the node configurations (refer to *Node Configurations*) at once, you should call the method *framework.node.Node.unfreeze_all()*.

4.1.2 Create Nodes with Low-Level Primitives

Instead of using the high-level API for describing a graph you can create it by using `fuddly` low-level primitives. Generally, you don't need to go through that, but for specific complex situations it could provide you with what you need. To create a graph or a single node, you always have to instantiate the class *framework.node.Node* which enables you to set the type of content for the main node configuration (refer to *Node Configurations*).

Depending on the content type the constructor will call the following methods to do the job:

- *framework.node.Node.set_values()*: for *typed-value* nodes.
- *framework.node.Node.set_subnodes_basic()*: for *non-terminal* nodes without specifying a grammar.
- *framework.node.Node.set_subnodes_with_csts()*: for *non-terminal* nodes constrained by a grammar.

- *framework.node.Node.set_generator_func()*: for *generator* nodes.
- *framework.node.Node.set_func()*: for *function* nodes.

Note: Methods specific to the node content (*framework.node.NodeInternals*) can be called directly on the node itself and it will be *forwarded* to the content (if the method name does not match one the *framework.node.Node* class).

See also:

If you want to learn more about the specific operations that can be performed on each kind of content (whose base class is *framework.node.NodeInternals*), refer to the related class, namely:

- *framework.node.NodeInternals_TypedValue*
- *framework.node.NodeInternals_NonTerm*
- *framework.node.NodeInternals_GenFunc*
- *framework.node.NodeInternals_Func*

4.1.3 Cloning a Node

A graph or any node within can be cloned in order to be used anywhere else independently from the original node. To perform such an operation you should use *framework.node.Node.get_clone()* like in the following example:

```
rnode_copy = rnode.get_clone('mycopy')
```

`rnode_copy` is a clone of the root node of the previous graph example, and as such it is a clone of the graph. The same operation can be achieved by creating a new node and passing as a parameter the node to copy:

```
rnode_copy = Node('mycopy', base_node=rnode, new_env=True)
```

When you clone a node you may want to keep its current state or ignore it (that is, cloning an unfrozen graph as if it was reset). For doing so, you have to use the parameter `ignore_frozen_state` of the method *framework.node.Node.get_clone()*. By default it is set to `False` which means that the state is preserved during the cloning process.

4.1.4 Display a Frozen Graph

If you want to display a frozen graph (representing one data) in ASCII-art you have to call the method *framework.node.Node.show()* on it. For instance the following:

```
rnode.show()
```

will display a frozen graph that looks the same as the one below:

```
[0] ex [NonTerm]
 \__(1) ex/data0 [String] size=4B
 |         \_raw: 'Plip'
 \__[1] ex/data_group [NonTerm]
 |    \__[2] ex/data_group/len [GenFunc | node_args: ex/data_group/data1]
 |    |    \__(3) ex/data_group/len/cts [UINT8] size=1B
 |    |              \_ 5 (0x5)
 |    |              \_raw: '\x05'
 |    \__(2) ex/data_group/data1 [String] size=5B
 |    |         \_raw: 'Test!'
 |    \__(2) ex/data_group/data2 [UINT16_be] size=2B
 |    |         \_ 40953 (0x9FF9)
 |    |         \_raw: '\x9f\xf9'
 |    \__(2) ex/data_group/data2:2 [UINT16_be] size=2B
 |    |         \_ 33836 (0x842C)
 |    |         \_raw: '\x84,'
 |    \__(2) ex/data_group/data3 [UINT8] size=1B
 |    |         \_ 40 (0x28)
 |    |         \_raw: '('
 |    \__(2) ex/data_group/data3:2 [UINT8] size=1B
 |              \_ 50 (0x32)
 |              \_raw: '2'
 \__(1) ex/data4 [String] size=3B
          \_raw: 'Red'
```

4.1.5 The Node Environment

The environment which should normally be the same for all the nodes of a same graph are handled by the following methods:

- *framework.node.Node.set_env()*
- *framework.node.Node.get_env()*

4.2 Search for Nodes in a Graph

Searching a graph for specific nodes can be performed in basically two ways. Depending on the criteria based on which you want to perform the search, you should use:

- *framework.node.Node.iter_nodes_by_path()*: iterator that walk through all the nodes that match the *graph path*—you provide as a parameter—from the node on which the method is called (or `None` if nothing is found). The syntax defined to represent paths is similar to the one of filesystem paths. Each path are represented by a python string, where node names are separated by `/`'s. For instance the path from the root node of the previous data model to the node named `len` is:

  ```
  'ex/data_group/len'
  ```

  Note the path provided is interpreted as a regexp.

- *framework.node.Node.get_first_node_by_path()*: use the previous iterator to provide the first node that match the *graph path* or `None` if nothing is found
- *framework.node.Node.get_reachable_nodes()*: It is the more flexible primitive that enables to perform a search based on syntactic and/or semantic criteria. It can take several optional parameters to define your search like a *graph path* regexp. Unlike the previous method it always returns a list, either filled with the nodes that has been found or with nothing. You can use other kinds of criteria to be passed through the following parameters:

- `internals_criteria`: To be provided with a *framework.node.NodeInternalsCriteria* object. This object enable you to describe the syntactic properties you look for, such as:
  - The node kind (refer to *Operations on Node Properties and Attributes*) and/or subkind (for a typed terminal node, a subkind is the class of its embedded typed value);
  - The node attributes (refer to *Operations on Node Properties and Attributes*)
  - The node constraints such as: *existence* or *quantity synchronization*. Usable constraints are defined by *framework.node.SyncScope*.
- `semantics_criteria`: To be provided with a `framework.node.NodeSemanticCriteria` object. This object enable you to describe the semantic properties you look for. They are currently limited to a list of python strings.
- `owned_conf`: The name of a node configuration (refer to *Node Configurations*) that the targeted nodes own.

Note: If the search is only path-based, *framework.node.Node.iter_nodes_by_path()* is the preferable solution as it is more efficient.

The following code snippet illustrates the use of such criteria for retrieving all the nodes coming from the `data2` description (refer to *Entangled Nodes*):

```
from framework.plumbing import *
from framework.node import *
from framework.value_types import *

fmk = FmkPlumbing()
fmk.start()

fmk.run_project(name='tuto')

ex_node = fmk.dm.get_atom('ex')
ex_node.freeze()

ic = NodeInternalsCriteria(mandatory_attrs=[NodeInternals.Mutable],
                           node_kinds=[NodeInternals_TypedValue],
                           negative_node_subkinds=[String],
                           negative_csts=[SyncScope.Qty])

sc = NodeSemanticsCriteria(mandatory_criteria=['sem1', 'sem2'])

ex_node.get_reachable_nodes(internals_criteria=ic, semantics_criteria=sc,
                            owned_conf='alt2')
```

Obviously, you don't need all these criteria for retrieving such node. It's only for exercise.

Note: For abstracting away the data model from the rest of the framework, `fuddly` uses the specific class *framework.data.Data* which acts as a data container. Thus, while interacting with the different part of the framework, Node-based data (or string-based data) should be encapsulated in a *framework.data.Data* object.

For instance `Data(ex_node)` will create an object that encapsulate `ex_node`. Accessing the node again is done through the property *framework.data.Data.content*

4.3 The Node Dictionary Interface

The *framework.node.Node* implements the dictionary interface, which means the following operation are possible on a node:

```
node[key]           # reading operation

node[key] = value   # writing operation
```

As a `key`, you can provide:

- A *path regexp* (where the node on which the method is called is considered as the root) to the nodes you want to reach. A list of the nodes will be returned for the reading operation (or `None` if the path match nothing), and for the writing operation all the matching nodes will get the new value. The reading operation is equivalent to calling *framework.node.Node.iter_nodes_by_path()* on the node and providing the parameter `path_regexp` with your path (except the method will return a python generator instead of a list).

  The following python code snippet illustrate the access to the node named `len` to retrieve its byte string representation:

  ```
  rnode['ex/data_group/len'][0].to_bytes()

  # same as:
  rnode.get_first_node_by_path('ex/data_group/len').to_bytes()
  ```

- A *framework.node.NodeInternalsCriteria* that match the internal attributes of interest of the nodes you want to retrieve and which are reachable from the current node. It is equivalent to calling *framework.node.Node.get_reachable_nodes()* on the node and providing the parameter `internals_criteria` with your criteria object. A list will always be returned, either empty or containing the nodes of interest.
- A *framework.node.NodeSemanticsCriteria* that match the internal attributes of interest of the nodes you want to retrieve and which are reachable from the current node. It is equivalent to calling *framework.node.Node.get_reachable_nodes()* on the node and providing the parameter `semantics_criteria` with the criteria object. A list will always be returned, either empty or containing the nodes of interest.

See also:

To learn how to create criteria objects refer to *Search for Nodes in a Graph*.

As a `value`, you can provide:

- A *framework.node.Node*: In this case the method *framework.node.Node.set_contents()* will be called on the node with the *node* as parameter.
- A *framework.node.NodeSemantics*: In this case the method *framework.node.Node.set_semantics()* will be called on the node with the *semantics* as parameter.
- A python integer: In this case the method `framework.value_types.INT.set_raw_values()` of the *INT* object embedded in the targeted node will be called with the *integer* as parameter. (Have to be only used with typed-value nodes embedding an `INT`.)
- A byte string: In this case the method *framework.node.Node.absorb()* will be called on the node with the *byte string* as parameter.

Warning: These methods should generally be called on a frozen graph.

4.4 Change a Node

You can change the content of a specific node by absorbing a new content (refer to *Byte String Absorption*).

You can also temporarily change the node value of a terminal node (until the next time *framework.node.Node.unfreeze()* is called on it) with the method *framework.node.Node.set_frozen_value()* (refer to *Generate Data a.k.a. Freeze a Graph*).

But if you want to make some more disruptive change and change a terminal node to a non-terminal node for instance, you have two options. Either you do it from scratch and you leverage the function described in the section *Create Nodes with Low-Level Primitives*. For instance:

```
node_to_change.set_values(value_type=String(max_sz=10))
```

Or you can do it by replacing the content of one node with another one. That allows you for instance to add a data from a model to another model. To illustrate this possibility let's consider the following code that change the node `data0` of our data model example with an USB `STRING` descriptor (yes, that does not make sense, but you can do it if you like ;).

```
from framework.plumbing import *

fmk = FmkPlumbing()
fmk.run_project(name='tuto')

usb_str = fmk.dm.get_external_atom(dm_name='usb', data_id='STR')

ex_node = fmk.dm.get_atom('ex')

ex_node['ex/data0'] = usb_str  # Perform the substitution

ex_node.show()                        # Data.show() will call .show() on the embedded node
```

The result is shown below:

Note: Releasing constraints (like a CRC, an offset, a length, ...) of an altered data can be useful if you want `fuddly` to automatically recomputes the constraint for you and still comply to the model. Refer to *Generate Data a.k.a. Freeze a Graph*.

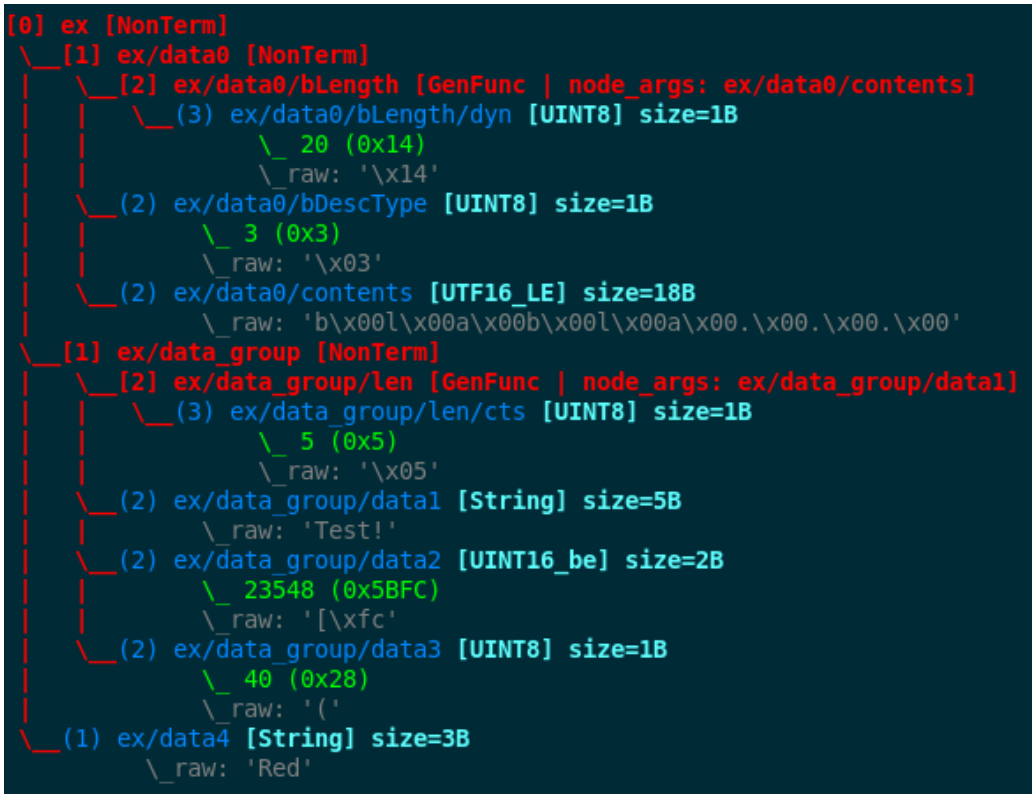
You can also add subnodes to non-terminal nodes through the usage of *framework.node.NodeInternals_NonTerm.add()*. For instance the following code snippet will add a new node after the node `data2`.

```
data2_node = ex_node['ex/data_group/data2'][0]
ex_node['ex/data_group$'][0].add(Node('my_node', values=['New node added']),
                                 after=data2_node)
```

Thus, if `ex_node` before the modification is:

```
[0] ex [NonTerm]
 \__(1) ex/data0 [String] size=4B
 |        \_ codec=iso8859-1
 |        \_raw: b'Plip'
 \__[1] ex/data_group [NonTerm]
 |   \__[2] ex/data_group/len [GenFunc | node_args: ex/data_group/data1]
 |   |   \__(3) ex/data_group/len/cts [UINT8] size=1B
```

(continues on next page)



(continued from previous page)

```
|   |           \_ 5 (0x5)
|   |           \_raw: b'\x05'
|   \__(2) ex/data_group/data1 [String] size=5B
|   |       \_ codec=iso8859-1
|   |       \_raw: b'Test!'
|   \__(2) ex/data_group/data2 [UINT16_be] size=2B
|   |       \_ 10 (0xA)
|   |       \_raw: b'\x00\n'
|   \__(2) ex/data_group/data3 [UINT8] size=1B
|           \_ 30 (0x1E)
|           \_raw: b'\x1e'
\__(1) ex/data4 [String] size=3B
        \_ codec=iso8859-1
        \_raw: b'Red'
```

After the modification it will be:

```
[0] ex [NonTerm]
 \__(1) ex/data0 [String] size=4B
 |       \_ codec=iso8859-1
 |       \_raw: b'Plip'
 \__[1] ex/data_group [NonTerm]
 |   \__[2] ex/data_group/len [GenFunc | node_args: ex/data_group/data1]
 |   |   \__(3) ex/data_group/len/cts [UINT8] size=1B
 |   |           \_ 5 (0x5)
 |   |           \_raw: b'\x05'
```

(continues on next page)

(continued from previous page)

```
|   \__(2) ex/data_group/data1 [String] size=5B
|   |        \_ codec=iso8859-1
|   |        \_raw: b'Test!'
|   \__(2) ex/data_group/data2 [UINT16_be] size=2B
|   |        \_ 10 (0xA)
|   |        \_raw: b'\x00\n'
|   \__(2) ex/data_group/my_node [String] size=14B
|   |        \_ codec=iso8859-1
|   |        \_raw: b'New node added'
|   \__(2) ex/data_group/data3 [UINT8] size=1B
|            \_ 30 (0x1E)
|            \_raw: b'\x1e'
\__(1) ex/data4 [String] size=3B
         \_ codec=iso8859-1
         \_raw: b'Red'
```

4.4.1 Operations on Node Properties and Attributes

The following methods enable you to retrieve the kind of content of the node. The provided answer is for the current configuration (refer to *Node Configurations*) if the `conf` parameter is not provided:

- *framework.node.Node.is_nonterm()*
- *framework.node.Node.is_typed_value()*
- *framework.node.Node.is_genfunc()*
- *framework.node.Node.is_func()*

Checking if a node is frozen (refer to *Generate Data a.k.a. Freeze a Graph*) can be done thanks to the method:

- *framework.node.Node.is_frozen()*

The following methods enable you to change specific node properties or attributes:

- Methods related to the keyword `fuzz_weight` described in the section *Data Model Keywords*:
  - *framework.node.Node.set_fuzz_weight()*
  - *framework.node.Node.get_fuzz_weight()*
- Methods related to the keywords `determinist`, `random`, `finite` and `infinite` described in the section *Data Model Keywords*:
  - *framework.node.Node.make_determinist()*
  - *framework.node.Node.make_random()*
  - *framework.node.Node.make_finite()*
  - *framework.node.Node.make_infinite()*
- Methods to deal with node attributes and related to the keywords `set_attrs` and `clear_attrs` described in the section *Data Model Keywords*:
  - *framework.node.Node.set_attr()*
  - *framework.node.Node.clear_attr()*
  - *framework.node.Node.is_attr_set()*

You can test the compliance of a node with syntactic and/or semantic criteria with the method *framework.node.Node.compliant_with()*. Refer to the section *Search for Nodes in a Graph* to learn how to specify criteria.

Any object can be added to a node as a private attribute. The private object should support the `__copy__` interface. To set and retrieve a private object the following methods are provided:

- *framework.node.Node.set_private()*
- *framework.node.Node.get_private()*

Node semantics can be defined to view the data model in a specific way, which boils down to be able to search for nodes based on semantic criteria (refer to *Search for Nodes in a Graph*). To set semantics on nodes or to retrieve them the following methods have to be used:

- *framework.node.Node.set_semantics()*: Take a list of strings (that capture the semantic) or a *framework.node.NodeSemantics*
- *framework.node.Node.get_semantics()*: Returns a *framework.node.NodeSemantics*

4.4.2 Node Configurations

Alternative node content can be added dynamically to any node of a graph. This is called a *node configuration* and everything that characterize a node—its type: non-terminal, terminal, generator; its attributes; its links with other nodes; and so on—are included within. A node is then a receptacle for an arbitrary number of *configurations*.

Note: When a node is created it gets a default configuration named `MAIN`.

Configuration management is based on the following methods:

- *framework.node.Node.add_conf()*: To add a new configuration.
- *framework.node.Node.remove_conf()*: To remove a configuration based on its name.
- *framework.node.Node.is_conf_existing()*: To check a configuration existence based on its name.
- *framework.node.Node.set_current_conf()*: To change the current configuration of a node with the one whose the name is provided as a parameter.
- *framework.node.Node.get_current_conf()*: To retrieve the name of the current node configuration.
- *framework.node.Node.gather_alt_confs()*: to gather all configuration names defined in the subgraph where the root is the node on which the method is called.

In what follows, we illustrate some node configuration change based on our data model example

```
rnode.freeze()   # We consider there is at least 2 'data2' nodes

# We change the configuration of the second 'data2' node
rnode['ex/data_group/data2:2'][0].set_current_conf('alt1', ignore_entanglement=True)
rnode['ex/data_group/data2:2'][0].unfreeze()

rnode.show()

# We change back 'data2:2' to the default configuration
rnode['ex/data_group/data2:2'][0].set_current_conf('MAIN', ignore_entanglement=True)
# We change the configuration of the first 'data2' node
rnode['ex/data_group/data2'][0].set_current_conf('alt1', ignore_entanglement=True)
# This time we unfreeze directly the parent node
```

(continues on next page)

(continued from previous page)

```
rnode['ex/data_group$'][0].unfreeze(dont_change_state=True)

rnode.show()
```

See also:

Refer to *Entangled Nodes* about the parameter `ignore_entanglement`.

If you want to act on a specific configuration of a node without changing first its configuration, you can leverage the `conf` parameter of the methods that support it. For instance, all the methods used for setting the content of a node (refer to *Create Nodes with Low-Level Primitives*) are *configuration aware*.

Note: If you need to access to the node internals (*framework.node.NodeInternals*) the following attributes are provided:

- *framework.node.Node.cc*: to access to the node internals of the current configuration.
- *framework.node.Node.c*: dictionary to access to the node internals of any configuration based on their name.

4.4.3 Node Corruption Infrastructure

You can also leverage the *Node-corruption Infrastructure* (based on hooks within the code) for handling various corruption types easily. This infrastructure is especially used by the generic disruptor `tSTRUCT` (refer to *Generic Disruptors*). This infrastructure is based on the following primitives:

- *framework.node.Env.add_node_to_corrupt()*
- *framework.node.Env.remove_node_to_corrupt()*

The typical way to perform a corruption with this infrastructure is illustrated in what follows. This example performs a corruption that changes from the model the allowed amount for a specific node (`targeted_node`) of a graph (referenced by `rnode`) that can be created during the data generation from the graph.

```
mini = 8
maxi = 10
rnode.env.add_node_to_corrupt(targeted_node, corrupt_type=Node.CORRUPT_NODE_QTY,
                              corrupt_op=lambda x, y: (mini, maxi))

corrupt_rnode = Node(rnode.name, base_node=rnode, ignore_frozen_state=False,
                     new_env=True)
rnode.env.remove_node_to_corrupt(targeted_node)
```

From now on, you have still a clean graph referenced by `rnode`, and a corrupted one referenced by `corrupt_rnode`. You can now instantiate some data from `corrupt_rnode` that complies to an altered data model (because we change the grammar that constrains the data generation).

The corruption operations currently defined are:

- *framework.node.Node.CORRUPT_NODE_QTY*
- *framework.node.Node.CORRUPT_QTY_SYNC*
- *framework.node.Node.CORRUPT_EXIST_COND*

4.4.4 Byte String Absorption

This feature is described in the tutorial. Refer to *Absorption of Raw Data that Complies to the Data Model* to learn about it. The methods which are involved in this process are:

- *framework.node.Node.absorb()*
- *framework.node.Node.set_absorb_helper()*
- *framework.node.Node.enforce_absorb_constraints()*

4.5 Miscellaneous Primitives

- *framework.node.Node.get_path_from()*: if it exists, return the first path to this node from the node provided as parameter; otherwise return None.
- *framework.node.Node.get_all_paths_from()*: similar as the previous method, except it returns a list of all the possible paths.

4.6 Entangled Nodes

Node descriptors that contain the `qty` attribute may trigger the creation of multiple nodes based on the same description. These nodes are created in a specific way to make them react as a group. We call the nodes of such a group: `entangled nodes`. If you perform a modification on any one node of the group (by calling a *setter* on the node for instance), all the other nodes will be affected the same way.

Some node methods are immune to the entanglement, especially all the *getters*, others enable you to temporarily break the entanglement through the parameter `ignore_entanglement`.

CHAPTER

FIVE

DATA ANALYSIS

Each data you send and all the related information (the way the data has been built, the feedback from the target, and so on) are stored within the `fuddly` database (an SQLite database located at `<fuddly data folder>/fmkdb.db`). They all get a unique ID, starting from 1 and increasing by 1 each time a data is sent.

5.1 FmkDB Toolkit

To interact with the database a convenient toolkit is provided (`<root of fuddly>/tools/fmkdb.py`).

5.1.1 Usage Examples

Let's say you want to look at all the information that have been recorded for one of the data you sent, with the ID 4. The following command will display a synthesis of what you want:

```
./tools/fmkdb.py -i 4
```

And if you want to get all information, issue the following:

```
./tools/fmkdb.py -i 4 --with-data --with-fbk
```

You can also request information on all data sent between two dates. For instance the following command will display all data information that have been recorded between 25th January 2016 (11:30) and 26th January 2016:

```
./tools/fmkdb.py --info-by-date 2016/01/25-11:30 2016/01/26
```

For further information refer to the help by issuing:

```
./tools/fmkdb.py -h
```

5.1.2 Fmkdb Toolkit Manual

Hereunder is shown the output of `<root of fuddly>/tools/fmkdb.py -h`.

```
usage: fmkdb.py [-h] [--fmkdb PATH] [--no-color] [-v] [--page-width WIDTH]
                [--fbk-src FEEDBACK_SOURCES] [--project PROJECT_NAME]
                [--fbk-status-formula STATUS_REF] [-s] [-i DATA_ID]
                [--info-by-date START END] [-ids FIRST_DATA_ID LAST_DATA_ID] [-wf] [-wd]␣
↪[-wa]
```

(continues on next page)

(continued from previous page)

```
                [--without-fmkinfo] [--without-analysis] [--limit LIMIT] [--raw] [-dd]
                [-df] [--data-atom ATOM_NAME] [--fbk-atom ATOM_NAME]
                [--force-fbk-decoder DATA_MODEL_NAME]
                [--export-data FIRST_DATA_ID LAST_DATA_ID] [-e DATA_ID]
                [--remove-data FIRST_DATA_ID LAST_DATA_ID] [-r DATA_ID]
                [--data-with-impact] [--data-with-impact-raw] [--data-without-fbk]
                [--data-with-specific-fbk FEEDBACK_REGEXP] [-a IMPACT COMMENT]
                [--disprove-impact FIRST_ID LAST_ID]

Argument for FmkDB toolkit script

optional arguments:
  -h, --help            show this help message and exit

Miscellaneous Options:
  --fmkdb PATH          Path to an alternative fmkDB.db
  --no-color            Do not use colors
  -v, --verbose         Verbose mode
  --page-width WIDTH    Width hint for displaying information

Configuration Handles:
  --fbk-src FEEDBACK_SOURCES
                        Restrict the feedback sources to consider (through a regexp).
                        Supported by: --data-with-impact, --data-without-fbk, --data-
↪with-
                        specific-fbk
  --project PROJECT_NAME
                        Restrict the data to be displayed to a specific project.␣
↪Supported
                        by: --info-by-date, --info-by-ids, --data-with-impact, --data-
                        without-fbk, --data-with-specific-fbk
  --fbk-status-formula STATUS_REF
                        Restrict the data to be displayed to specific feedback status.
                        This option provides the formula to be used for feedback status
                        filtering (the character "?" should be used in place of the␣
↪status
                        value that will be checked). Supported by: --data-with-impact

Fuddly Database Visualization:
  -s, --all-stats       Show all statistics

Fuddly Database Information:
  -i DATA_ID, --data-id DATA_ID
                        Provide the data ID on which actions will be performed. Without
                        any other parameters the default action is to display information
                        on the specified data ID.
  --info-by-date START END
                        Display information on data sent between START and END (date
                        format 'Year/Month/Day' or 'Year/Month/Day-Hour' or
                        'Year/Month/Day-Hour:Minute')
  -ids FIRST_DATA_ID LAST_DATA_ID, --info-by-ids FIRST_DATA_ID LAST_DATA_ID
                        Display information on all the data included within the specified
```

(continues on next page)

(continued from previous page)

```
                        data ID range
  -wf, --with-fbk       Display full feedback (expect --data-id)
  -wd, --with-data      Display data content (expect --data-id)
  -wa, --with-async-data
                        Display any related async data (expect --data-id)
  --without-fmkinfo     Do not display fmkinfo (expect --data-id)
  --without-analysis    Do not display user analysis (expect --data-id)
  --limit LIMIT         Limit the size of what is displayed from the sent data and the
                        retrieved feedback (expect --with-data or --with-fbk).
  --raw                 Display data and feedback in raw format

Fuddly Decoding:
  -dd, --decode-data    Decode sent data based on the data model used for the selected
                        data ID or the atome name provided by --atom
  -df, --decode-fbk     Decode feedback based on the data model used for the selected␣
↪data
                        ID or the atome name provided by --fbk-atom
  --data-atom ATOM_NAME
                        Atom of the data model to be used for decoding the sent data. If
                        not provided, the name of the sent data will be used.
  --fbk-atom ATOM_NAME  Atom of the data model to be used for decoding feedback. If not
                        provided, the default data model decoder will be used (if one
                        exists), or the name of the first registered atom in the data
                        model
  --force-fbk-decoder DATA_MODEL_NAME
                        Decode feedback with the decoder of the data model specified

Fuddly Database Operations:
  --export-data FIRST_DATA_ID LAST_DATA_ID
                        Extract data from provided data ID range
  -e DATA_ID, --export-one-data DATA_ID
                        Extract data from the provided data ID
  --remove-data FIRST_DATA_ID LAST_DATA_ID
                        Remove data from provided data ID range and all related
                        information from fmkDB
  -r DATA_ID, --remove-one-data DATA_ID
                        Remove data ID and all related information from fmkDB

Fuddly Database Analysis:
  --data-with-impact    Retrieve data that negatively impacted a target. Analysis is
                        performed based on feedback status and user analysis if present
  --data-with-impact-raw
                        Retrieve data that negatively impacted a target. Analysis is
                        performed based on feedback status
  --data-without-fbk    Retrieve data without feedback
  --data-with-specific-fbk FEEDBACK_REGEXP
                        Retrieve data with specific feedback provided as a regexp
  -a IMPACT COMMENT, --add-analysis IMPACT COMMENT
                        Add an impact analysis to a specific data ID (expect --data-id).
                        IMPACT should be either 0 (no impact) or 1 (impact), and COMMENT
                        provide information
  --disprove-impact FIRST_ID LAST_ID
```

(continues on next page)

(continued from previous page)

```
                        Disprove the impact of a group of data present in the outcomes of
                        '--data-with-impact-raw'. The group is determined by providing␣
↪the
                        smaller data ID (FIRST_ID) and the bigger data ID (LAST_ID).
```

5.2 Plotty

Plotty is a tool used to vizualize data from the fmkDB. To interact with the database a convenient toolkit is provided (<root of fuddly>/tools/plotty/*).

5.2.1 Usage Examples

A very common usage is just to plot the data relatively to the date it was sent at. To do that, you can use the plotty CLI, at <root of fuddly>/tools/plotty/plotty.py:

```
./tools/plotty.py -ids '0..100|2'
```

Plots the SEND_DATE in function of the ID of every message which has en even ID beetween 0 and 100. A lot of display and formatting options are available to build your own plotting experience

For further information refer to the help by issuing:

```
./tools/plotty/plotty.py -h
```

5.2.2 Plotty Manual

Hereunder is shown the output of <root of fuddly>/tools/plotty/plotty.py -h

```
usage: plotty.py [-h] -ids ID_RANGE [-df DATE_FORMAT] [-db PATH [PATH ...]] [-f FORMULA]␣
↪[-poi POINTS_OF_INTEREST] [-gm {all,poi,auto}] [-hp]
           [-l ANNOTATIONS [ANNOTATIONS ...]] [-al ASYNC_ANNOTATIONS [ASYNC_ANNOTATIONS .
↪..]] [-o OTHER_ID_RANGE] [-s VERTICAL_SHIFT]

Arguments for Plotty

options:
  -h, --help            show this help message and exit

Main parameters:
  -ids ID_RANGE, --id-range ID_RANGE
                        The ID range to take into account should be: either <id_start>..
↪<id_stop>[|<step>], or <id_start_1>..<id_stop_1>[|<step_1>], ...,
                        <id_start_n>..<id_stop_n>[|<step_n>]
  -df DATE_FORMAT, --date-format DATE_FORMAT
                        Wanted date format, in a strftime format (1989 C standard).␣
↪Default is %H:%M:%S.%f
  -db PATH [PATH ...], --fmkdb PATH [PATH ...]
                        Path to any fmkDB.db files. There can be many if using the --
↪other_id_range option. Default is fuddly/data/directory/fmkDB.db
```

(continues on next page)

(continued from previous page)

```
Display Options:
  -f FORMULA, --formula FORMULA
                        The formula to plot, in the form "y ~ x"
  -poi POINTS_OF_INTEREST, --points-of-interest POINTS_OF_INTEREST
                        How many point of interest the plot should show. Default is none
  -gm {all,poi,auto}, --grid-match {all,poi,auto}
                        Should the plot grid specifically match some element. Possible␣
↪options are 'all', 'poi' and 'auto'. Default is 'all'
  -hp, --hide-points    Should the graph display every point above the line, or just the␣
↪line. Default is to display the points

Labels Configuration:
  -l ANNOTATIONS [ANNOTATIONS ...], --labels ANNOTATIONS [ANNOTATIONS ...]
                        Display the specified labels for each Data ID represented in the␣
↪curve. ('t' for TYPE, 'g' for TARGET, 's' for SIZE, 'a' for
                        ACK_DATE)
  -al ASYNC_ANNOTATIONS [ASYNC_ANNOTATIONS ...], --async-labels ASYNC_ANNOTATIONS [ASYNC_
↪ANNOTATIONS ...]
                        Display the specified labels for each Async Data ID represented␣
↪in the curve. ('i' for 'ID', 't' for TYPE, 'g' for TARGET, 's' for
                        SIZE)

Multiple Curves Options:
  -o OTHER_ID_RANGE, --other-id-range OTHER_ID_RANGE
                        Other ranges of IDs to plot against the main one. All other␣
↪options apply to it
  -s VERTICAL_SHIFT, --vertical-shift VERTICAL_SHIFT
                        When --other-id-range is used, specify the spacing between the␣
↪curves. The shift is computed as the multiplication between the
                        original curve height and this value
```

5.2.3 Concrete output example

Given a simple command line:

```
./tools/plotty/plotty.py -id '2..12' -l t -al t i -df "%M:%S.%f" -poi 3
```

It is already possible to vizualize trends, behaviors and anomalies. Comparison becomes easier !

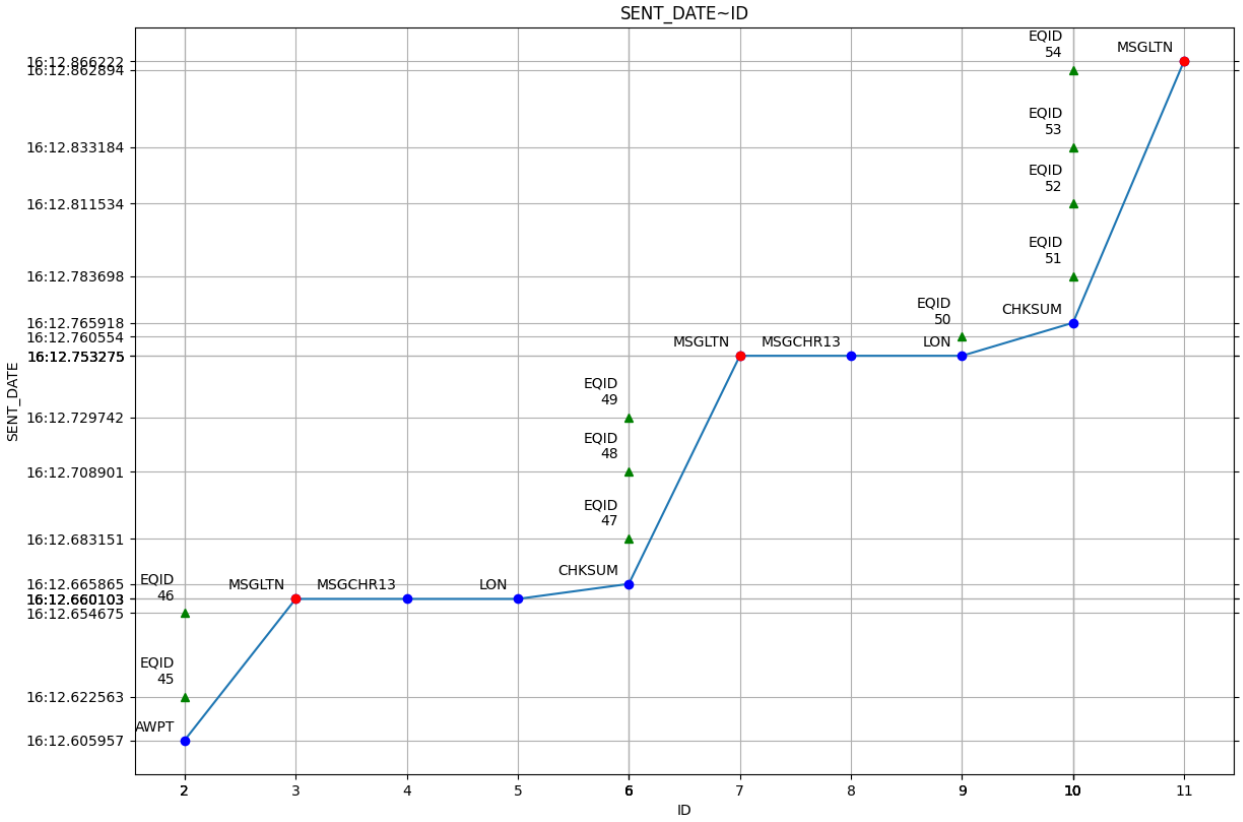


Fig. 1: Example of curve comparison using Plotty

CHAPTER

SIX

SCENARIO INFRASTRUCTURE

6.1 Overview

The *Scenario Infrastructure* enables you to describe protocols which are based on the data described in a data model. You can do whatever you want, either by following a standard or playing around it.

Once a *scenario* has been defined and registered (refer to *Scenario Description*), `Fuddly` will automatically create a specific *Generator* that comply to what you described.

Note: The `Fuddly` shell command `show_dmaker_types` displays all the data maker, *Generators* and *Disruptors*. The Generators which are backed by a scenario are prefixed by `SC_`.

A *scenario* is a state-machine. Its description follows an oriented graph where the nodes, called *steps*, define the data to be sent to the target. The transitions that interconnect these steps can be guarded by different kinds of callbacks that trigger at different moment (e.g., after sending the data, or after having retrieved any feedback from the target or any probes registered to monitor the target).

Finally, once a scenario has been described, some automatic alterations can be performed on it. It is especially useful in the case of protocol fuzzing to assess the robustness of the target regarding protocol sequencing, timing, and so on. This is described in the section *Scenario Fuzzing*.

6.2 Scenario Description

6.2.1 A First Example

Let's begin with a simple example that interconnect 3 steps in a loop without any callback.

Note: All the examples (or similar ones) of this chapter are provided in the file `<fuddly_root>/data_models/tutorial/tuto_strategy.py`.

```
1  from framework.tactics_helpers import Tactics
2  from framework.scenario import *
3
4  tactics = Tactics()
5
6  periodic1 = Periodic(Data('Periodic (5s)\n'), period=5)
7  periodic2 = Periodic(Data('One Shot\n'), period=None)
```

(continues on next page)

(continued from previous page)

```
8
9  step1 = Step('exist_cond', fbk_timeout=1, set_periodic=[periodic1, periodic2],
10              do_before_sending=before_sending_cbk, vtg_ids=0)
11 step2 = Step('separator', fbk_timeout=2, clear_periodic=[periodic1], vtg_ids=1)
12 step3 = NoDataStep(clear_periodic=[periodic2])
13 step4 = Step('off_gen', fbk_timeout=0)
14
15 step1.connect_to(step2)
16 step2.connect_to(step3, cbk_after_fbk=check_answer)
17 step3.connect_to(step4)
18 step4.connect_to(step1, cbk_after_sending=check_switch)
19
20 sc1 = Scenario('ex1', anchor=step1, user_context=UI(switch=False))
21
22 tactics.register_scenarios(sc1)
```

Note that scenarios can be described:

- either in a `*_strategy.py` file that matches the data model you base your scenarios on;
- or in a project file, if your scenarios use different data models involved in your project or the scenarios are agnostic to any data model but have a meaning only at project level.

In what follows we illustrate how to describe scenarios in the context of the data model `mydf` defined in `tuto.py` (refer to *Defining the Imaginary MyDF Data Model* for further explanation on file organization). Describing scenarios in the context of a project will be the same except for the scenarios registration step, where the method *framework.project.Project.register_scenarios()* will have to be used to make them available within the framework.

In our example, the registration goes through the special object `tactics` (line 4) of `tuto_strategy.py` which is usually used to register the data makers (*disruptors* or *generators*) specific to a data model (refer to *Defining Specific Disruptors* for details), but also used to register scenarios as shown in line 20.

From line 9 to 13 we define 4 *framework.scenario.Step*:

- The first one commands the framework to send a data of type `exist_cond` (which is the name of a data registered in the data model `mydf`) as well as starting 2 periodic tasks (threaded entities of the framework) that will emit each one a specific data. The first one will send the specified string every 5 seconds while the other one will send another string only once. Additionaly, the callback `before_sending_cbk` is set and will be triggered when the framework will reach this step (callbacks are discussed in a later section). Note that the step sets also the maximum time duration that `Fuddly` should respect for collecting the feedback from the target (feedback timeout). This timeout is actually handled by the `Target` object, which may decide to respect it or not. For instance the `NetworkTarget` respect it while the `EmptyTarget` (default target) do not. Note that the feedback mode (refer to *Generic Targets*) is also supported and can be set through the parameter `fbk_mode`. Finally, the parameter `vtg_ids` is used to allows interacting in a multi-targets environment (this topic is detailed in *Scenario Involving Multiple Targets*). In this case it asks to send the data to the target referenced by the virtual ID 0.
- The second step commands the framework to send a data of type `separator` and change the feedback timeout to 2. Additionally, it requests the framework to stop the first periodic task, and asks it to send its data to the target referenced by the virtual ID 1.
- The third step do nothing except requesting the framework to stop the second periodic task.
- The fourth step requests to send a data of type `off_gen` and change back the feedback timeout to 0. Additionally it commands the framework to stop the periodic task which is currently running.

Note: The feedback timeout will directly influence the time that seperates the execution of each step

The linking of these steps is carried out from the line 15 to 18. Some callbacks are defined and are explained in a later section. Then in line 20, a *framework.scenario.Scenario* object is created with the name `ex1` which is used by `Fuddly` for naming the *generator* that implements this scenario. It prefixes it with the string `SC_` leading to the name `SC_EX1`. The *scenario* is then linked to the initial *step* in line 18.

Note: The `user_context` parameter of the Scenario class used in line 20 allows to provide parameters to Steps and callbacks of the scenario (through the `ScenarioEnv` object shared between them and described in a later section).

This parameter can be filled with any object. Anyway, the preferable object class to use is `framework.global_resources.UI` which is the container class also used to pass parameters to `Generators` and `Disruptors`.

The execution of this scenario will follow the pattern:

```
step1 --------------------> step2 ---------> step3 -------> step1 ---------> ...
  |                           |                |              |
  \--> periodic1 ...  [periodic1 stopped]      |              \--> periodic1 ...
  \--> periodic2 ...                   [periodic2 stopped]    \--> periodic2 ...
```

You can play with this scenario by loading the `tuto` project with the `TestTarget` 7 and 8 (useful to provide arbitrary feedback):

```
[fuddly term]>> run_project tuto 7 8
[fuddly term]>> send_loop 10 SC_EX1
```

If you want to visualize your scenario, you can issue the following command (`[FMT]` is optional and can be `xdot`, `pdf`, `png`, ...):

```
[fuddly term]>> show_scenario SC_EX1 [FMT]
```

If you want to monitor the current step of the scenario each time you trigger the generator that runs through it, you have to set the generic parameter `graph` to `True`. Then, each time you trigger the generator the current step will be shown in blue:

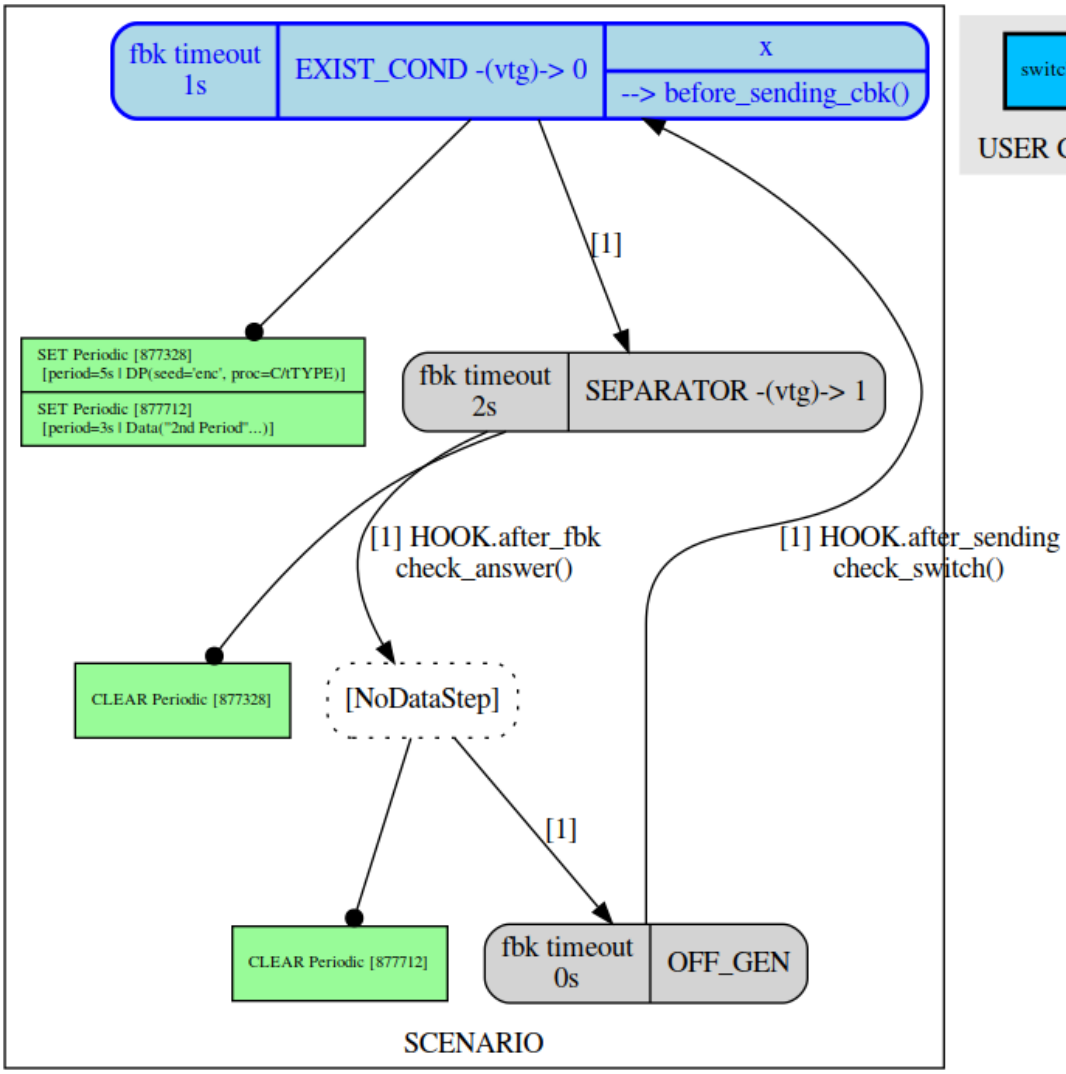
```
[fuddly term]>> send SC_EX1(graph=True:graph_format=xdot)
```

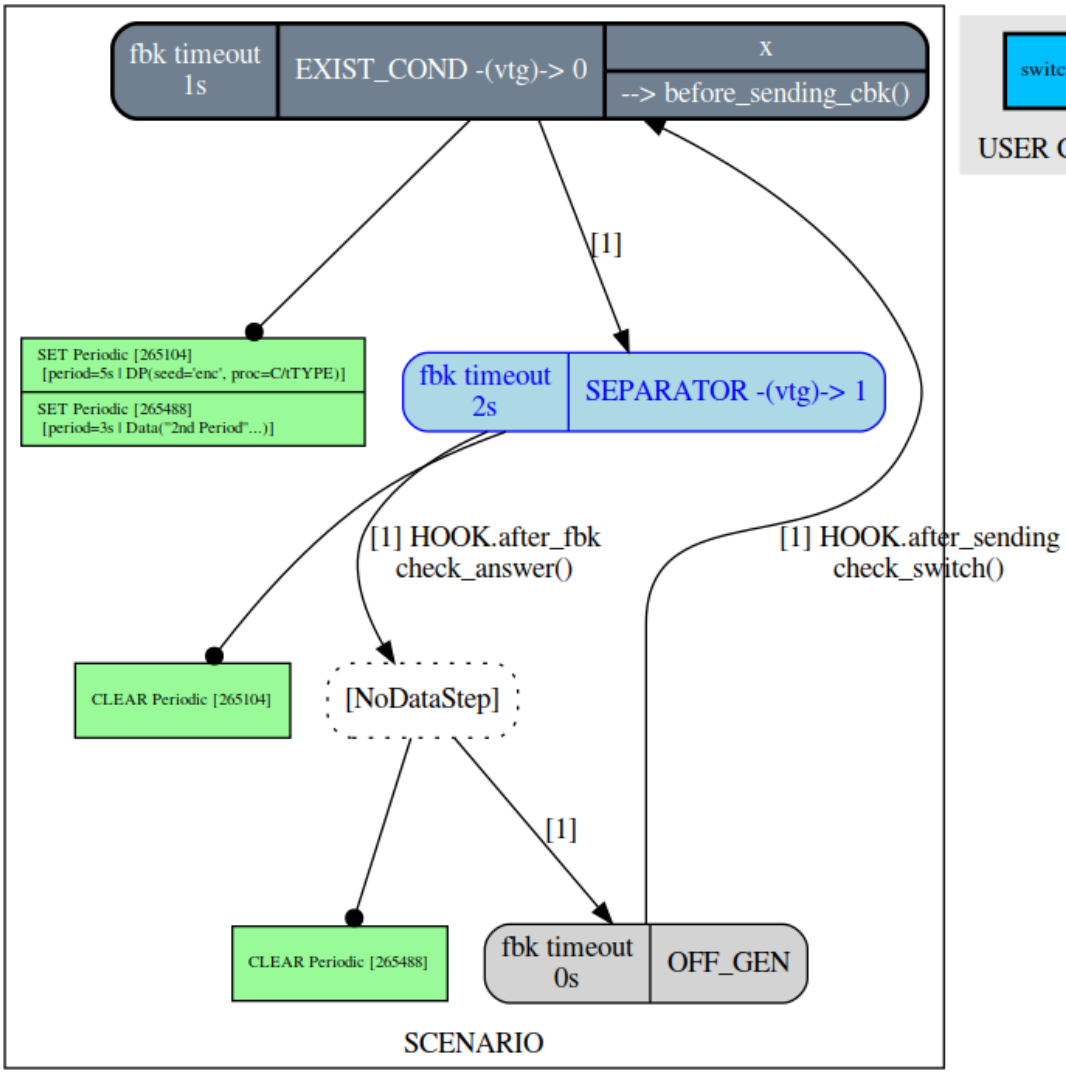
Then after another call:

```
[fuddly term]>> send SC_EX1(graph=True:graph_format=xdot)
```

Note: All available parameters can be consulted by issuing the following command (like any data generators):

```
[fuddly term]>> show_generators SC_EX1
```


fbk timeout
1s
EXIST_COND -(vtg)-> 0
x
--> before_sending_cbk()
switch = False
USER CONTEXT
[1]
SET Periodic [877328]
[period=5s | DP(seed='enc', proc=C/tTYPE)]
SET Periodic [877712]
[period=3s | Data("2nd Period"...)]
fbk timeout
2s
SEPARATOR -(vtg)-> 1
[1] HOOK.after_fbk
check_answer()
[1] HOOK.after_sending
check_switch()
CLEAR Periodic [877328]
[NoDataStep]
[1]
CLEAR Periodic [877712]
fbk timeout
0s
OFF_GEN
SCENARIO


fbk timeout
1s
EXIST_COND -(vtg)-> 0
X
--> before_sending_cbk()
switch = False
USER CONTEXT
[1]
SET Periodic [265104]
[period=5s | DP(seed='enc', proc=C/tTYPE)]
SET Periodic [265488]
[period=3s | Data("2nd Period"...)]
fbk timeout
2s
SEPARATOR -(vtg)-> 1
[1] HOOK.after_fbk
check_answer()
[1] HOOK.after_sending
check_switch()
CLEAR Periodic [265104]
[NoDataStep]
[1]
CLEAR Periodic [265488]
fbk timeout
0s
OFF_GEN
SCENARIO

6.2.2 Steps

The main objective of a *framework.scenario.Step* is to command the generation and sending of one or multiple data to targets selected in the framework. The data generation depends on what has been provided to the parameter `data_desc` of a *framework.scenario.Step*. This is described in the section *Data Generation Process*.

Note that the data generated in one step will be sent by default to the first loaded target. If the scenario you describe involve different targets, you could then refer to them by specifying virtual target IDs in the step constructor thanks to the parameter `vtg_ids`. Virtual target IDs are then to be mapped to real targets within the project file. Refer to *Scenario Involving Multiple Targets*.

A step can also modify the way the feedback is handled after the data have been emitted by the framework. The parameters `fbk_timeout`, and `fbk_mode` (refer to *Generic Targets*) are used for such purpose and are applied to the current target (by the framework) when the step is reached.

A step can additionally triggers the execution of periodic tasks that will emit some user-specified data (note the execution will trigger after feedback retrieval from the framework). This is done by providing a list of *framework.scenario.Periodic* to the parameter `set_periodic`. And, in order to stop previously started periodic tasks, the parameter `clear_periodic` have to be filled with a list of references on the relevant periodic tasks.

See also:

Refer to the section *A First Example* for practical information on how to use such features.

A step can also start a periodic or a one-shot task whose content could be entirely specified by the user. This is done by providing a list of *libs.utils.Task* to the parameter `start_tasks`. And, in order to stop previously started periodic tasks, the parameter `stop_tasks` have to be filled with a list of references on the relevant periodic tasks. Details on tasks are provided here *Defining Tasks*.

In addition to the features provided by a step, some user-defined callbacks can be associated to a step and executed while the framework is handling the step (that is generating data as specified by the step and sending it):

- If some code need to be executed when a step is reached and before any data is processed from it, you can leverage the parameter `do_before_data_processing` of the *framework.scenario.Step* class. It has to be provided with a function satisfying the following signature:

  ```
  1  def before_data_generation_cbk(env, step)
  ```

  where `step` is a reference to the *framework.scenario.Step* on which the action is executed, and `env` is a reference to the scenario environment *framework.scenario.ScenarioEnv*.

- And if some code need to be executed within a step after data has been processed and just before its sending, you can leverage the parameter `do_before_sending` of the *framework.scenario.Step* class. It has to be provided with a function satisfying the following signature:

  ```
  1  def before_sending_cbk(env, step)
  ```

  where the parameters have the same meaning as previously.

Note also that a step once executed will display a description related to what it did. You can override this description by providing the `step_desc` parameter of a *framework.scenario.Step* constructor with a python string.

Finally, some subclasses of *framework.scenario.Step* have been defined to make a scenario description easier:

- *framework.scenario.FinalStep*: When such kind of step is reached, it terminates the execution of the scenario. It is equivalent to a `Step` with its `final` attribute set to `True`.
- *framework.scenario.NoDataStep*: This kind of step should be used when the purpose is not to generate and send data but only to use other step features (e.g., feedback timeout or mode). Besides, the step callback `do_before_data_processing` will still be triggered if some code need to be executed (but

do_before_sending will not). And all the transitions from this step would only trigger their callback cbk_after_fbk to evaluate their condition.

6.2.3 Transitions

When two steps are connected together thanks to the method *framework.scenario.Step.connect_to()* some callbacks can be specified to perform any user-relevant action before crossing the transition that links up the two steps, but also to decide if this transition can be crossed. They act as transition conditions.

Indeed, a callback has to return *True* if it wants the framework to cross the transition, otherwise it should return *False*. If no callback is defined the transition is considered to be not guarded and thus can be crossed without restriction. Besides, only one transition is chosen at each step. It is the first one, by order of registration, that can be activated (at least one callback that returns *True*, or no callback at all). It is worth noting that the transitions are executed in a minimalistic way, meaning that if a callback return *True*, the associated transition will be chosen and no other callback will be executed (except all the callbacks from the selected transition) before a next step need to be selected.

Two types of callback can be associated to a transition through the parameters cbk_after_sending and cbk_after_fbk of the method *framework.scenario.Step.connect_to()*. A brief explanation is provided below:

cbk_after_sending To provide a function that will be executed before the execution of the next step, and just after the sending of the data from the current step. Its signature is as follows:

```
def callback(scenario_env, current_step, next_step)
```

The current_step is the one that is in progress and which is connected to next_step by the transition containing the current callback. The scenario_env parameter is a reference to the scenario environment *framework.scenario.ScenarioEnv*, which is shared between all the steps and transitions of a scenario.

Note: A scenario environment *framework.scenario.ScenarioEnv* provides some information like an attribute dm which is initialized with the *framework.data_model.DataModel* related to the scenario; or an attribute target which is initialized with the current target in use (a subclass of framework.target.Target).

A scenario environment can also be used as a shared memory for all the steps and transitions of a scenario.

---

cbk_after_fbk To provide a function that will be executed before the execution of the next step, and just after Fuddly retrieved the feedback of the target (and/or any registered probes). Its signature is as follows:

```
def callback(scenario_env, current_step, next_step, feedback)
```

This type of callback takes the additional parameter feedback filled by the framework with the target and/or probes feedback further to the current step data sending. It is an object *framework.database.FeedbackGate* that provides the handful method *framework.database.FeedbackGate.iter_entries()* which returns a generator that iterates over:

- all the feedback entries associated to a specific feedback source provided as a parameter—and for each entry the triplet (status, timestamp, content) is provided;
- all the feedback entries if the source parameter is None—and for each entry the 4-uplet (source, status, timestamp, content) is provided. Note that for such kind of iteration, the *framework.database.FeedbackGate* object can also be directly used as an iterator—avoiding a call to *framework.database.FeedbackGate.iter_entries()*.

This object can also be tested as a boolean object, returning False if there is no feedback at all.

Note that a callback can modify a step. For instance, considering an imaginary protocol, and after sending a registration request to a network service (initial step), feedback from the target are provided to the callbacks registered on the next transitions. These callbacks could then look for an identifier within the feedback and then update the next step to make it sending a message with the right identifier.

A step has a property `node` that provides the root node (*framework.node.Node*) of the modeled data it contains or *None* if the data associated to the step is a raw data (like `Data('raw data')`). Any callback can then alter the `node` of a step in order to update it with usefull information. In our example, the `node` is updated with the identifier (refer to line 10-11 of the following code snippet).

Note: Accessing to `next_step.content` from a callback will provide *None* in the case the next step include a raw data. In the case it includes a `DataProcess`, `next_step.content` will provide the *framework.node.Node* corresponding to the `DataProcess`'s `seed` or `None` (if no seed is available or the seed is raw data). In the latter case, the data process would not have been carried out at the time of the callback execution, hence the `None` value. (Refer to the section *Data Generation Process*)

Note: You can leverage the dissection/absorption mechanism of `Fuddly` to deal with the feedback if you have modeled the responses of the target. Refer to *Absorption of Raw Data that Complies to the Data Model* for further explanation on that matter.

Another aspect of callbacks is the ability to prevent the framework from going on (that is sending further data, and walking through the scenario) until a condition has been reached (related to the target feedback for instance). For that purpose, the callback needs to call the method `make_blocked()` on the current step and to return *False*. In this case, the callback `cbk_after_fbk` will be (re)called after the feedback gathering time has elapsed once again. Note that you can *block* from any callback, but only `cbk_after_fbk` will be called further on and will be able to *unblock* the situation.

Such ability can be useful if you are not sure about the time to wait for the answer of a network service for instance. This is illustrated in the following example in the lines 2-4.

```
1  def feedback_callback(env, current_step, next_step, feedback):
2      if not feedback:
3          # While no feedback is retrieved we stay at this step
4          current_step.make_blocked()
5          return False
6      else:
7          # Extract info from feedback and add an attribute to the scenario env
8          env.identifier = handle_fbk(feedback)
9          current_step.make_free()
10         if next_step.content:
11             next_step.content['off_gen/prefix'] = env.identifier
12         return True
13
14 periodic1 = Periodic(Data('1st Periodic (5s)\n'), period=5)
15 periodic2 = Periodic(Data('2nd Periodic (3s)\n'), period=3)
16
17 step1 = Step('exist_cond', fbk_timeout=2, set_periodic=[periodic1, periodic2])
18 step2 = Step('separator', fbk_timeout=5)
19 step3 = NoDataStep()
20 step4 = Step(DataProcess(process=[('C', UI(nb=1)),'tTYPE'], seed='enc'))
21
22 step1.connect_to(step2)
```

(continues on next page)

(continued from previous page)

```
23 step2.connect_to(step3, cbk_after_fbk=feedback_callback)
24 step3.connect_to(step4)
25 step4.connect_to(FinalStep())
26
27 sc2 = Scenario('ex2', anchor=step1)
```

In line 25 a *framework.scenario.FinalStep* (a step with its `final` attribute set to *True*) is used to terminate the scenario as well as all the associated periodic tasks that are still running. Note that if a callback set the `final` attribute of the `next_step` to *True*, it will trigger the termination of the scenario if this `next_step` is indeed the one that will be selected next.

Note: A step with its `final` attribute set to `True` will never trigger the sending of the data it contains.

Remark also the *framework.scenario.NoDataStep* in line 19 (`step3`) which is a step that does not provide data. Thus, the framework won't send anything during the execution of this kind of step. Anyway, it is still possible to set or clear some *periodic* in this step (or changing feedback timeout, …)

Note: A *framework.scenario.NoDataStep* is actually a step on which `make_blocked()` has been called on it and where `make_free()` do nothing.

The execution of this scenario will follow the pattern:

```
step1 --> step2 --> step2 ... step2 --> step3 --> step4 --> FinalStep()
  |              |                   |                          |
  |          No feedback          Feedback                      |
  |                                                             |
  \--> periodic1 ...                                   [periodic1 stopped]
  \--> periodic2 ...                                   [periodic2 stopped]
```

The last example illustrates a case where one step is connected to two other steps with a callback that rules the routing decision.

```
def routing_decision(env, current_step, next_step):
    if env.user_context.switch:
        return False
    else:
        env.user_context.switch = True
        return True

anchor = Step('exist_cond')
option1 = Step(Data('Option 1'))
option2 = Step(Data('Option 2'))

anchor.connect_to(option1, cbk_after_sending=routing_decision)
anchor.connect_to(option2)
option1.connect_to(anchor)
option2.connect_to(anchor)

sc3 = Scenario('ex3', anchor=anchor, user_context=UI(switch=False))
```

The execution of this scenario will follow the pattern:

```
anchor --> option1 --> anchor --> option2 --> anchor --> option2 --> ...
```

In addition to the callbacks, a transition can be guarded by booleans linked to specific conditions. They have to be specified as parameters of the method *framework.scenario.Step.connect_to()*. The current defined condition is:

- *DataProcess completed* (parameter is `dp_completed_guard`): which means, for a step hosting a *framework.data.DataProcess*, that if no more data can be issued by it the condition is satisfied, and thus the transition can be crossed. This is illustrated by the following example:

```
step1 = Step(DataProcess(process=['tTYPE'], seed='4tg1'))
step2 = Step(DataProcess(process=['tTYPE#2'], seed='4tg2'))

step1.connect_to(step2, dp_completed_guard=True)
step2.connect_to(FinalStep(), dp_completed_guard=True)

sc_proj3 = Scenario('proj3', anchor=step1)
```

6.2.4 Data Generation Process

The data produced by a *framework.scenario.Step* or a *framework.scenario.Periodic* is described by a *data descriptor* which can be:

- a python string referring to the name of a registered data from a data model;
- a *framework.data.Data*;
- a *framework.data.DataProcess*.

A *framework.data.DataProcess* is composed of a chain of generators and/or disruptors (with or without parameters) and optionally a `seed` on which the chain of disruptor will be applied to (if no generator is provided at the start of the chain).

A *framework.data.DataProcess* can trigger the end of the scenario if a disruptor in the chain yields (meaning it has terminated its job with the provided data: it is *exhausted*). If you prefer that the scenario goes on, then you have to set the `auto_regen` parameter to `True`. In such a case, when the step embedding the data process will be reached again, the framework will rerun the chain. This action will reset the exhausted disruptor and make new data available to it (by pulling data from preceding data makers in the chain or by using the *seed* again).

Additional *data maker chains* can be added to a *framework.data.DataProcess* thanks to *framework.data.DataProcess.append_new_process()*. Switching from the current process to the next one is carried out when the current one is interrupted by a yielding disruptor. Note that in the case the data process has its `auto_regen` parameter set to `True`, the current interrupted chain won't be rerun until every other chain has also get a chance to be executed.

See also:

Refer to *How to Perform Automatic Modification on Data* for more information on disruptor chaining.

Note: It follows the same pattern as the instructions that can set a virtual operator (*Defining Operators*). It is actually what the method *framework.plumbing.FmkPlumbing.process_data()* takes as parameters.

Here under examples of steps leveraging the different ways to describe their data to send.

```
Step( 'exist_cond' )   # 'exist_cond' is the name of a data from `mydf` data model

Step( Data('A raw message') )

Step( DataProcess(process=['ZIP', 'tSTRUCT', ('SIZE', UI(sz=100))]) )
Step( DataProcess(process=['C', 'tTYPE'], seed='enc') )
Step( DataProcess(process=['C'], seed=Data('my seed')) )
```

Steps may be configured to change the process of data generation. The following methods are defined for such purpose:

- *framework.scenario.Step.make_blocked()* and *framework.scenario.Step.make_free()*
- *framework.scenario.Step.set_dmaker_reset()* and *framework.scenario.Step.clear_dmaker_reset()*

Finally, it is possible for a `Step` to describe multiple data to send at once; meaning the framework will be ordered to use `framework.target.Target.send_multiple_data()` (refer to *Defining the Targets*). For that purpose, you have to provide the `Step` constructor with a list of *data descriptors* (instead of one).

6.2.5 Scenario Involving Multiple Targets

If you want to define a scenario that involves multiple targets, you will have to refer to the different targets through virtual target IDs. To illustrate such case, let's look at the `ex1` scenario defined in the `tuto` data model (refer to the file `data_models/tutorial/tuto_strategy.py`). `step1` and `step2` are defined with respectively the virtual target ID `0` and the virtual target ID 1:

```
step1 = Step(... vtg_ids=0)
step2 = Step(... vtg_ids=1)
```

Then, in order to use this scenario in your project you will have to provide a mapping with real targets thanks to the method *framework.project.Project.map_targets_to_scenario()*. For instance in the `tuto` project (refer to the file `projects/tuto_proj.py`), a mapping is created for the scenario `ex1`:

```
project.map_targets_to_scenario('ex1', {0: 8, 1: 9, None: 9})
```

A mapping is a simple python dictionnary that maps virtual target IDs to real target IDs. In our case, virtual IDs 0 and 1 have been mapped respectiveley to real IDs 8 and 9. Finally, the last association with the `None` virtual target ID is to cover data generated by steps that did not specify any virtual IDs at all.

6.3 Scenario Fuzzing

6.3.1 Overview

`Fuddly` implements different approaches to assess the robustness of a target with respect to its protocol handling, assuming a *framework.scenario.Scenario* has been defined to describe the protocol:

1. **Invert the transition conditions**: For each scenario step having guarded transitions, a new scenario is created where transition conditions are inverted. (Refer to *Invert Transition Conditions*.)

2. **Ignore the scenario timing constraints**: For each scenario step enforcing a timing constraint, a new scenario is created where any timeout conditions are removed (i.e., set to 0 second). (Refer to *Ignore Timing Constraints*.)

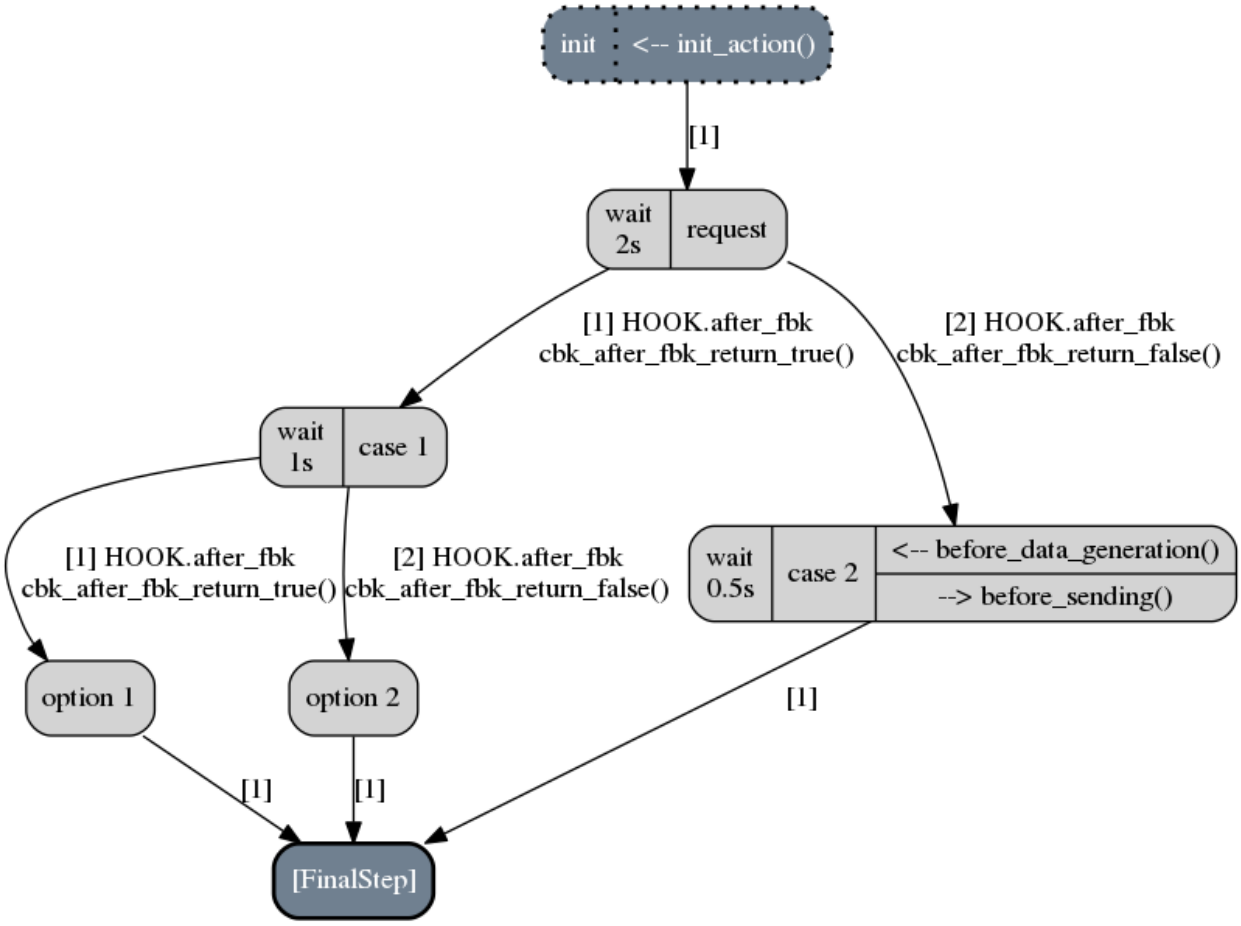
3. **Fuzz the data sent by the scenario**: For each scenario step that generates data, a new scenario is created where the data generated by the step is fuzzed. (Refer to *Fuzz the Data Sent by the Scenario*.)

4. **Make the protocol stutter**: For each scenario step that generates data, a new scenario is created where the step is altered to stutter a given number of times, meaning that data-sending steps would be triggered many times. (Refer to *Make the protocol stutter*.)

Note: The implemented approaches 1 and 2 can be used together, but they cannot be used in conjunction with the approach 3 or approach 4.

6.3.2 Fuzzing by Example

To illustrate the implemented fuzzing approaches let's take the following scenario representing an imaginary protocol.



Note: It is described by the following code snippet extracted from `data_models/tutorial/tuto_strategy.py`:

```
init = NoDataStep(step_desc='init', do_before_data_processing=init_action)
request = Step(Data(Node('request', vt=UINT8(values=[1, 2, 3]))),
               fbk_timeout=2)
case1 = Step(Data(Node('case 1', vt=String(values=['CASE 1']))),
             fbk_timeout=1)
case2 = Step(Data(Node('case 2', vt=String(values=['CASE 2']))),
             fbk_timeout=0.5,
```

(continues on next page)

(continued from previous page)

```
8               do_before_data_processing=before_data_generation,
9               do_before_sending=before_sending)
10  final_step = FinalStep()
11  option1 = Step(Data(Node('option 1', vt=SINT16_be(values=[10,15]))))
12  option2 = Step(Data(Node('option 2', vt=UINT8(min=3, max=9))))
13
14  init.connect_to(request)
15  request.connect_to(case1, cbk_after_fbk=cbk_after_fbk_return_true)
16  request.connect_to(case2, cbk_after_fbk=cbk_after_fbk_return_false)
17  case1.connect_to(option1, cbk_after_fbk=cbk_after_fbk_return_true)
18  case1.connect_to(option2, cbk_after_fbk=cbk_after_fbk_return_false)
19  case2.connect_to(final_step)
20  option1.connect_to(final_step)
21  option2.connect_to(final_step)
22
23  reinit = Step(Data(Node('reinit', vt=String(values=['REINIT']))))
24  reinit.connect_to(init)
25
26  sc_tuto_ex4 = Scenario('ex4', anchor=init, reinit_anchor=reinit)
```

Note the scenario does not depends on a data model definition, because it defines itself the data to send.

6.3.2.1 Invert Transition Conditions

If you want to invert the transition conditions of this scenario on a step-by-step basis (meaning that each step where transitions can be inverted will trigger the generation of a scenario altering the step while the other steps will remain untouched), you can issue the following command:

```
[fuddly term]>> send SC_EX4(cond_fuzz=True)
```

And if you want to display the scenario in `xdot` while running through it issue the following command instead:

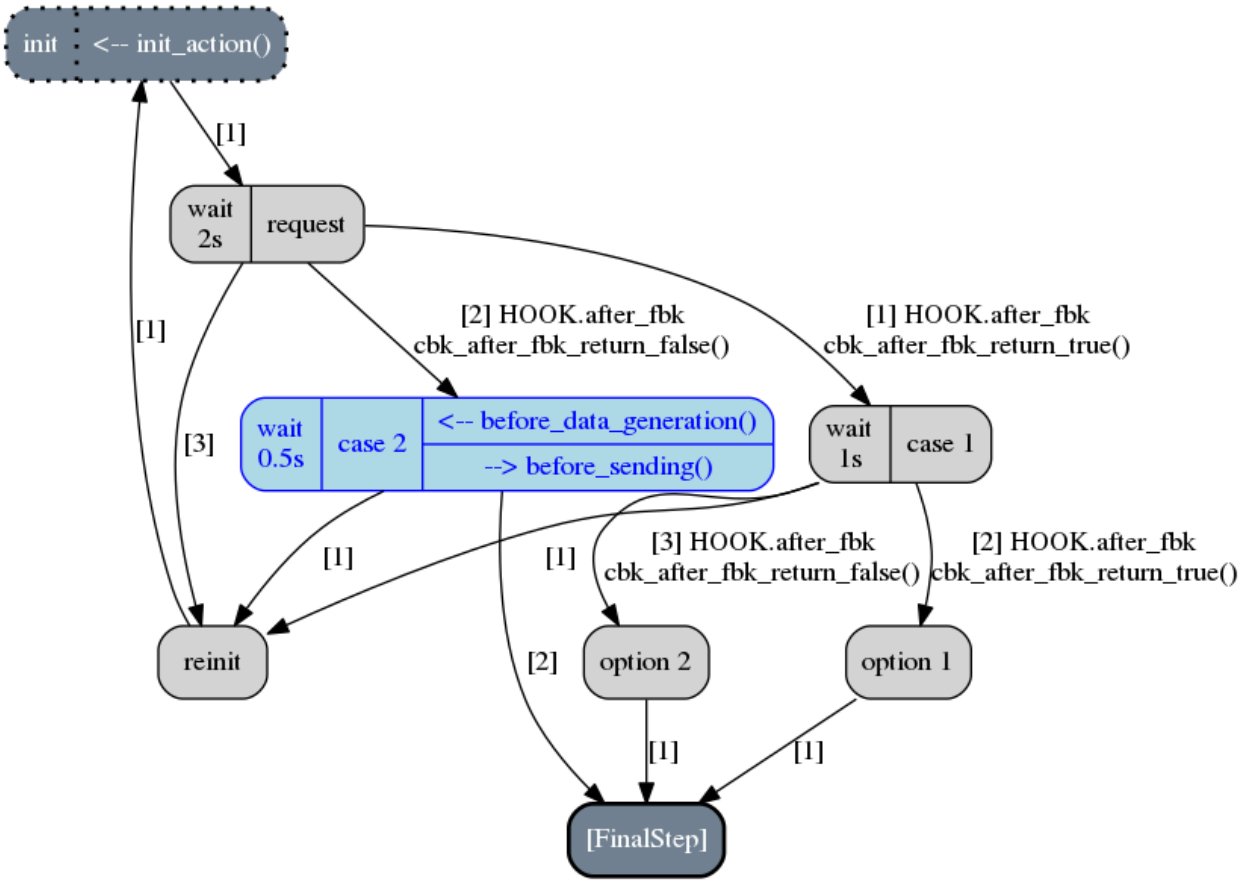
```
[fuddly term]>> send SC_EX4(cond_fuzz=True:graph=True:graph_format=xdot)
```

The result will be that the following scenario—altered version of the original one—will begin to run:

This picture represents the state of the scenario after three calls to the generator `SC_EX4` (where the current step is depicted in blue). Note that the first step that has been selected by `fuddly` for altering its transition is the `request` step (because the sole transition of the `init` step is not guarded). This alteration means that the conditions that guard the transitions of this step are inverted in order to alter the protocol logic. Practically, it means in our example that the `case 2` is chosen instead of the `case 1` because the transition condition that returns `False` on the original scenario, returns `True` in the altered one.

Note also that by default, after the alteration outcomes have been triggered (in this case after the `case 2` step has run), then a reinitialization sequence is initiated, in order to continue with the next altered scenario case. The reinitialisation sequence is by default a simple connection to the anchor of the scenario. But if the scenario has been provided with a reinitialization sequence (through the `reinit_anchor` parameter of *framework.scenario.Scenario*), this will be used instead. Our scenario example provide such reinitialization sequence (line 23-24 of the previous code snippet) and the picture depicts the use of it (all steps that follow the corrupted one are connected to it as well as the corrupted one in last resort if no transition can be crossed).

But if you don't want `fuddly` to perform a reinitialization after each alteration case, you simply have to set the generator parameter `reset` to `False`. In this case, the next alteration will trigger whenever the scenario will cross again the initial


init
<-- init_action()
[1]
wait
2s
request
[1]
[2] HOOK.after_fbk
cbk_after_fbk_return_false()
[1] HOOK.after_fbk
cbk_after_fbk_return_true()
[3]
wait
0.5s
case 2
<-- before_data_generation()
--> before_sending()
wait
1s
case 1
[1]
[1]
[3] HOOK.after_fbk
cbk_after_fbk_return_false()
[2] HOOK.after_fbk
cbk_after_fbk_return_true()
reinit
[2]
option 2
option 1
[1]
[1]
[FinalStep]

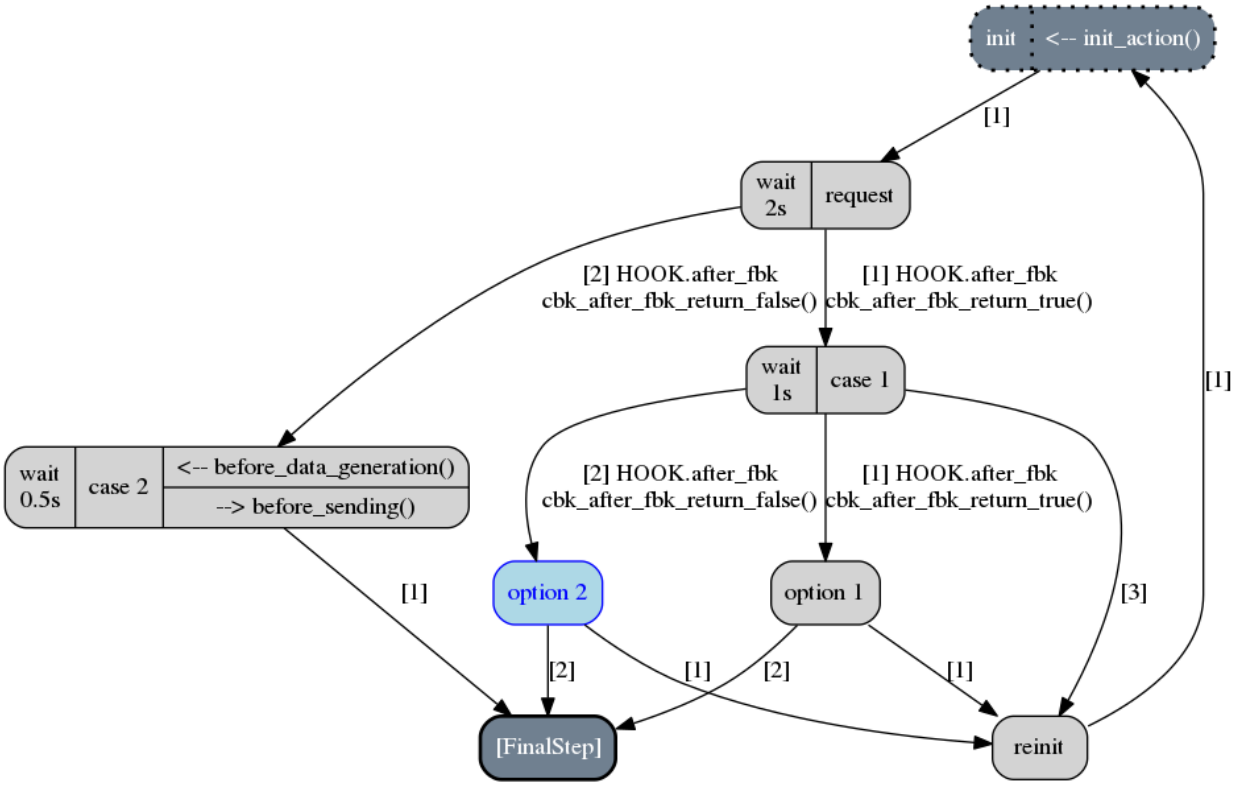
step (i.e., the scenario anchor), but only if it cross it (which may never happen depending on the scenario).

The following picture depicts the next altered scenario that will be instantiated if we continue to run through the generator `SC_EX4` (which has been configured with the option `cond_fuzz`).

Note: If you want to change the parameters of the generator while it is not exhausted, you need to reset it manually by issuing the following command:

```
>> reset_dmaker SC_EX4
```

For more details refer to *Resetting & Cloning Disruptors*



In this next altered scenario, the transition conditions of the `request` step are no more inverted. Thus, we have selected the `case 1` step has stated by the original scenario. But then, we choose the `option 2` step instead of the `option 1` because we inverted the conditions of the `case 1` step outgoing transitions.

Note: In some cases, some altered scenario cases may not terminate depending on the original scenario and the interaction with the evaluated target. To overcome such situation, you can stop a scenario whenever you want and then choose the next altered scenario case manually through the `init` parameter.

6.3.2.2 Ignore Timing Constraints

This approach follow the same pattern than (and is compatible with) the approach *Invert Transition Conditions*, (meaning that each step to be altered will trigger the generation of a scenario altering that step while the other steps will remain untouched). But instead of inverting the transition conditions, it generates cases that ignore timing constraints. To launch such alterations you can issue the following command:

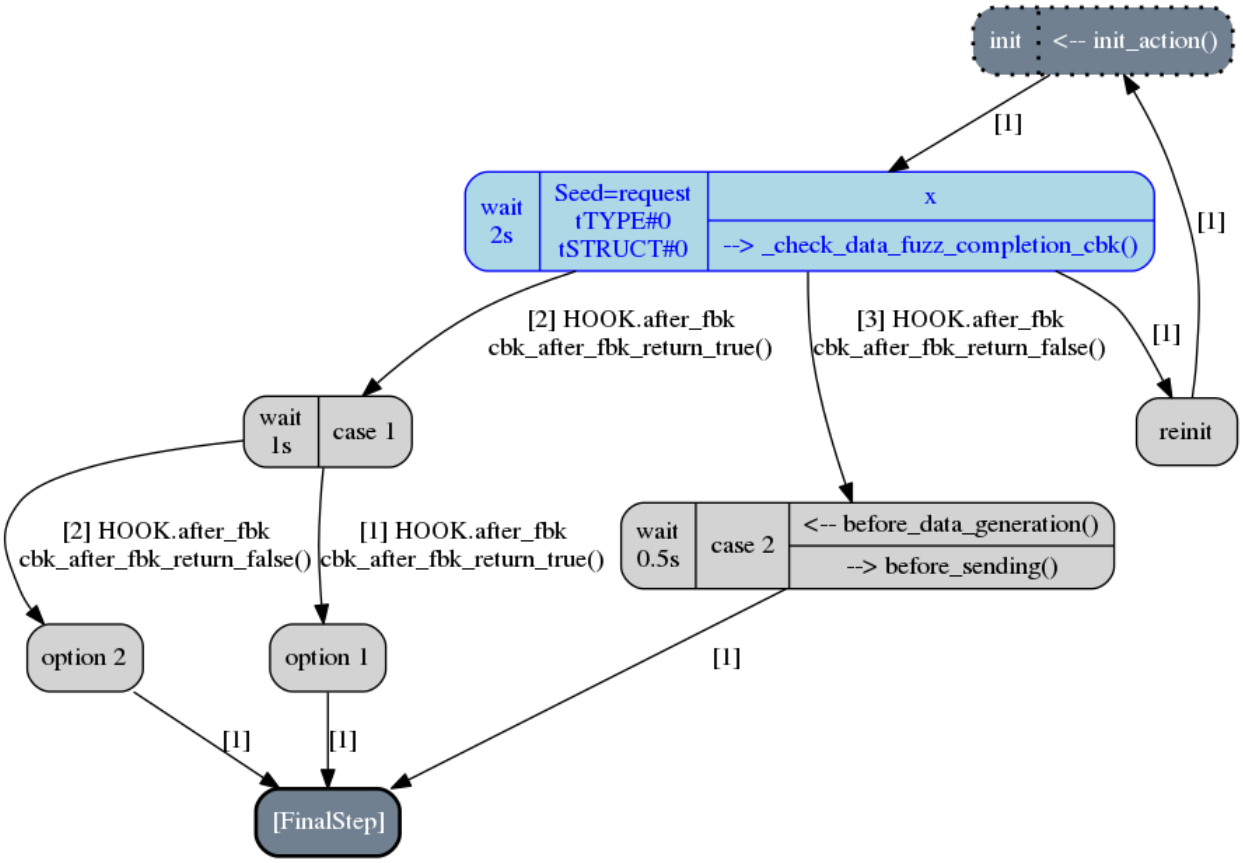
```
[fuddly term]>> send SC_EX4(ignore_timing=True)
```

6.3.2.3 Fuzz the Data Sent by the Scenario

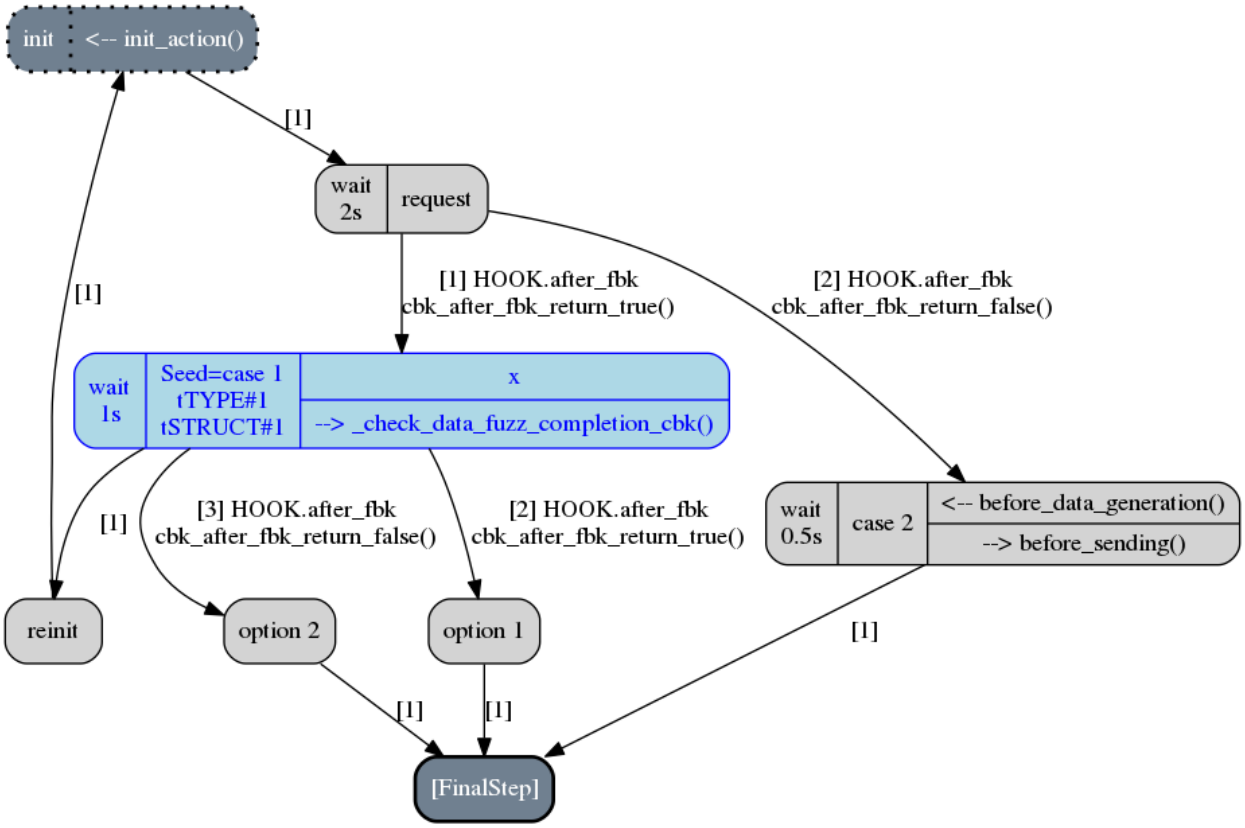
If you want to fuzz the data generated by the example scenario on a step-by-step basis, you can issue the following command

```
[fuddly term]>> send SC_EX4(data_fuzz=True)
```

This approach follow the same pattern than the approach *Invert Transition Conditions* (meaning that each step to be altered will trigger the generation of a scenario altering that step while the other steps will remain untouched). The follwing figure depicts the third call to the generator where the scenario run through the `request` step, but, contrary to the original scenario, some disruptors has been added, namely `tTYPE` and `tSTRUCT` (refer to *Generic Disruptors* for more information on them). Thus, instead of sending the correct `request` data, an altered version (handled firstly by `tTYPE`) will be sent. The scenario will then go back to the `init` step by taking the reinitialization path, in order to send the next altered data that `tTYPE` can produce with the `request` input. This loop will continue until the `tTYPE` disruptor exhausts, then the `tSTRUCT` disruptor will take over until exhaustion.



And finally, when all the alteration cases with the `request` step will be performed, the next altered scenario will be created. The following picture illustrate this case:



6.3.2.4 Make the protocol stutter

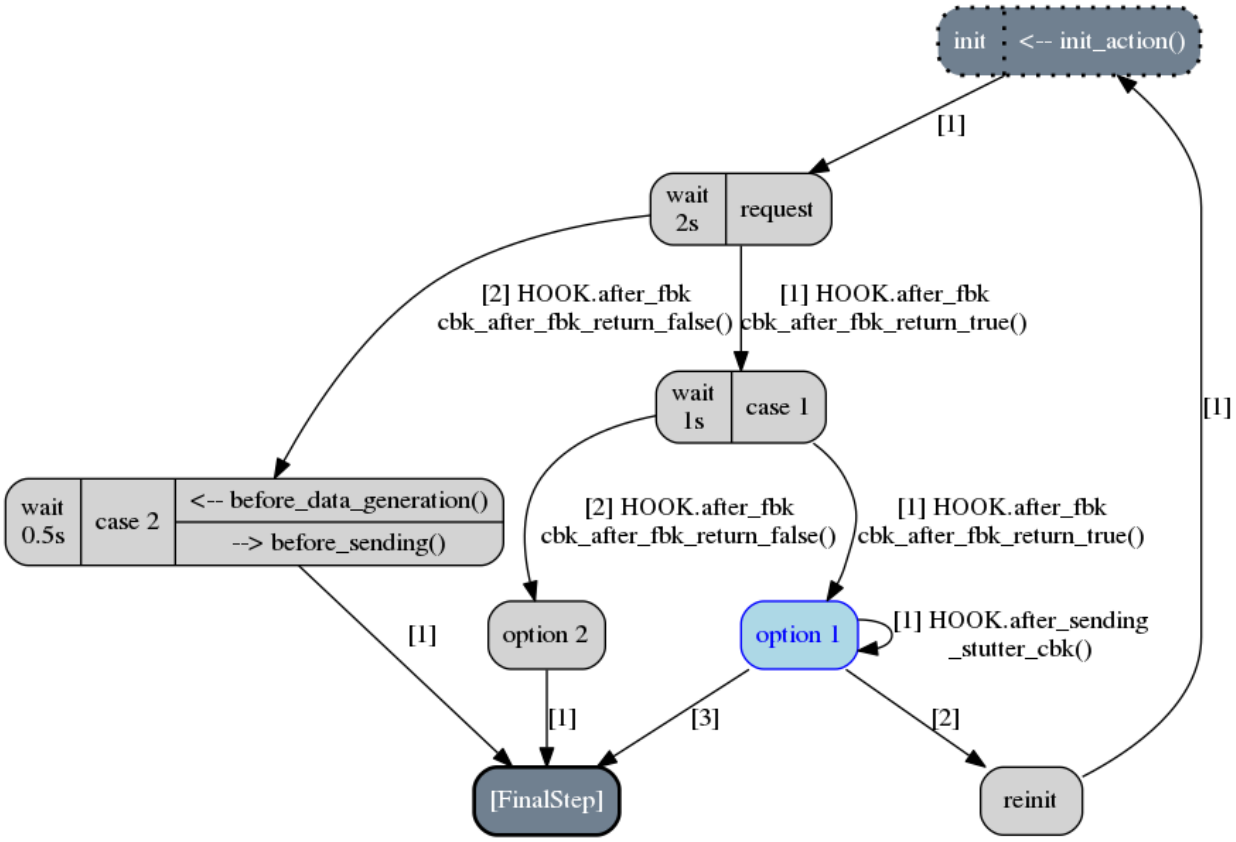
If you want to make stutter the data-sending steps of the example scenario you can issue the following command

```
[fuddly term]>> send SC_EX4(stutter=True)
```

You can also use the parameter `stutter_max` to specify the number of times a step have to stutter.

This approach follow the same pattern than the approach *Invert Transition Conditions* (meaning that each step to be altered will trigger the generation of a scenario altering that step while the other steps will remain untouched).

The following figure depicts the moment where the step which has been altered to stutter is `option 1`:


init
<-- init_action()
[1]
wait
2s
request
[2] HOOK.after_fbk
cbk_after_fbk_return_false()
[1] HOOK.after_fbk
cbk_after_fbk_return_true()
wait
1s
case 1
[1]
wait
0.5s
case 2
<-- before_data_generation()
--> before_sending()
[2] HOOK.after_fbk
cbk_after_fbk_return_false()
[1] HOOK.after_fbk
cbk_after_fbk_return_true()
[1]
option 2
option 1
[1] HOOK.after_sending
_stutter_cbk()
[1]
[3]
[2]
[FinalStep]
reinit

CHAPTER

SEVEN

KNOWLEDGE INFRASTRUCTURE

The *Knowledge Infrastructure* enables to:

- to dynamically collect feedback from Targets (*Generic Targets*) and Probes (*Generic Probes and Backend*), and extract information from it through dedicated handlers in order to create knowledge (refer to *Defining a Feedback Handler to Create Knowledge from Targets' and Probes' Feedback*);
- to add knowledge about the targets under test (e.g., the kind of OS, the used programming language, the hardware, and so on) in your project file (refer to *Adding Knowledge About the Targets Under Test in the Project File*);
- and to leverage this knowledge in relevant fuddly subsystems or in user-defined scenarios, disruptors, ... (refer to *Leveraging the Knowledge*). For instance, fuzzing a `framework.value_types.Filename` typed-node with the disruptor tTYPE will adapt the generated data relative to the OS, Language, and so on if this information is available.

7.1 Get Knowledge about the Targets Under Test

7.1.1 Defining a Feedback Handler to Create Knowledge from Targets' and Probes' Feedback

The `knwoledge.feedback_handler.FeedbackHandler` class provides the frame to create knowledge based on feedback retrieved by fuddly (essentially from targets themselves and the probes that monitor them).

In your projects, you can use already defined feedback handlers in order to automatically extract information from feedback and create knowledge that will be directly usable in various relevant fuddly components (refer to *Leveraging the Knowledge*).

Let's illustrate that with the `tuto` project (refer to `<fuddly_root>/projects/tuto_proj.py`) that register a `fuddly`-defined feedback handler whose sole purpose is to present the feature:

```
project.register_feedback_handler(TestFbkHandler())
```

This handler is defined as follows:

```
class TestFbkHandler(FeedbackHandler):

    def extract_info_from_feedback(self, source, timestamp, content, status):

        if content is None:
            return None
        elif b'Linux' in content:
            return OS.Linux
```

(continues on next page)

(continued from previous page)

```
9           elif b'Windows' in content:
10              return OS.Windows
```

It implements the method `knowledge.feedback_handler.FeedbackHandler.extract_info_from_feedback()` that is called each time feedback are retrieved from a target or a probe with parameters enabling you to process it, and return information about the target (in this case either `OS.Linux` or `OS.Windows`) if it can or `None` if it is not able.

The information concept is implemented through the class *framework.knowledge.information.Info*, and provide specific methods to increase or decrease the confidence that we have about a specific information. Each time a feedback handler return specific information like `OS.Linux` for instance, the framework would increase the confidence it has on it through the method *framework.knowledge.information.Info.increase_trust()*. Note that at any given time you can look at the current confidence level for any information by using the `framework.knowledge.information.Info.show_trust()` method.

The accumulation of information and the computed confidence level for each piece of it make up the knowledge on the targets under test.

If you want to look at the current state of the knowledge pool, you can issue the following command from the FmkShell:

```
>> show_knowledge
```

That will provide something similar to the following output:

```
-=[ Status of Knowledge ]=-

Info: Language.C [TrustLevel.Maximum --> value: 50]
Info: Hardware.X86_64 [TrustLevel.Maximum --> value: 50]
```

As dealing with feedback can be specific to your projects, you can obviously define your own feedback handlers for matching your specific needs. In order to do that you will have to create a new class that inherits from `knwoledge.feedback_handler.FeedbackHandler` and implements your specific behaviors. Then you will only need to register it in your *framework.project.Project* in order for its methods to be called automatically by fuddly at the relevant times.

Note: Even if initial purpose of feedback handlers is to create knowledge from retrieved information, it can be used to trigger other kinds of actions that fit your needs.

`knowledge.feedback_handler.FeedbackHandler` provides other methods that could be useful to overload to extract more information about the context of the feedback. Indeed, the method `knowledge.feedback_handler.FeedbackHandler.notify_data_sending()` is called each time data is sent and provide you with useful contextual information:

- the sent data;
- the date of emission;
- the targets.

7.1.2 Adding Knowledge About the Targets Under Test in the Project File

As seen in Section *Defining a Feedback Handler to Create Knowledge from Targets' and Probes' Feedback*, knowledge on the targets under test can be built upon the information extracted from feedback retrieved while interacting with the targets. But it can also be something known from the beginning. If you know you are dealing with a C program, and that program is executed on an x86 architecture, then you would like to provide this knowledge right ahead, so that fuddly could leverage them to optimize its fuzzing for instance.

In order to provide such knowledge, you simply have to call *framework.project.Project.add_knowledge()* in your project file with your knowledge on the targets.

```
project.add_knowledge(
    Hardware.X86_64,
    Language.C
)
```

7.1.3 Information Categories and How to Define More

The current information categories are:

- *framework.knowledge.information.OS*
- *framework.knowledge.information.Hardware*
- *framework.knowledge.information.Language*
- *framework.knowledge.information.InputHandling*

Depending on your project, you may want to define new specific information categories. In such case, You will simply have to define new python enumeration that inherits from *framework.knowledge.information.Info* in your project file. Then, you would need to use them in specific feedback handler (refer to *Defining a Feedback Handler to Create Knowledge from Targets' and Probes' Feedback*) in order to leverage them within specific scenarios or disruptors for instance.

7.2 Leveraging the Knowledge

7.2.1 Automatic Fuddly Adaptation to Knowledge

It is a work in progress.

Currently, data models that use the following node types within their description will benefit from knowledge about the targets under test:

- *framework.value_types.String*: Specific cases related to *framework.knowledge.information.Language* are added.
- *framework.value_types.Filename*: Specific cases related to *framework.knowledge.information.OS* and *framework.knowledge.information.Language* are added.

If knowledge on the targets are provided to the framework (either from the project file or because some in-use feedback handlers populated at some point the knowledge pool) then the previous type nodes will restrict their own fuzzing cases, impacting directly the disruptor `tTYPE` (refer to *tTYPE - Advanced Alteration of Terminal Typed Node*) in order to avoid doing irrelevant tests (e.g., providing a C format strings input to an ADA program).

If there is no knowledge on a specific category, then all specific fuzzing cases related to that category will still be provided.

7.2.2 Leveraging Knowledge in User-defined Components

Knowledge on the targets under tests can be used by various components of the framework and is made available to the user in various context like:

- Scenario specification (refer to *Scenario Infrastructure*) where all callbacks can access the knowledge pool through the scenario environment (`framework.scenario.ScenarioEnv`) under the attribute *knowledge_source*.
- Disruptors or generators implementation (refer to *Defining Specific Disruptors*), through the attribute `framework.tactics_helpers.DataMaker.knowledge_source`.
- Data model description (refer to *Data Modeling*), through the attribute `framework.data_model.DataModel.knowledge_source`.

These parameters refer to a global object defined for the project as a set of `framework.knowledge.information.Info`.

CHAPTER

EIGHT

EVOLUTIONARY FUZZING

8.1 Overview

Evolutionary fuzzing is a search technique inspired by evolutionary biology (Darwin) which aims at converging towards the discovery of weaknesses. It uses genetic algorithms in order to produce successive generations of test cases populations. The test cases creation is not only based on classic methods but also on the feedback retrieved from the targets. In this context, the first population is the only one instantiated the traditional way. The ones that follow are spawned using five steps:

1. **Fitness score computation**: each test case, member of the current population, is given a score which is function of some metrics (impact on the target, diversity, and so on), which are calculated by the entity in charge of the monitoring aspects.
2. **Probabilities of survival association**: depending on the score computed in the previous step, a probability of survival is associated to each individual.
3. **Dice are rolled**: using the probabilities of survival: weakest test cases are killed.
4. **Mutation**: aims to modify a little bit each individuals (flip some bits for instance) to find local optimums.
5. **Cross-over**: on the contrary, involves huge changes in order to find other optimums. It combines the test cases that are still alive in order to generate even better solutions. This process is also used to compensate the kills done in step 3.

The implementation within `Fuddly` is divided into three main components:

- A *framework.evolutionary_helpers.Population* class that is composed of *framework.evolutionary_helpers.Individual* instances. Each individual represents a data to be sent. The population, on the other hand, contains all the evolutionary logic. More details about these classes are given in the next section.
- The *framework.generic_data_makers.g_population* generator that loops through the population members and triggers an evolution when necessary.
- A scenario only used for one of its callback. It is in charge of retrieving the feedback after each sending.

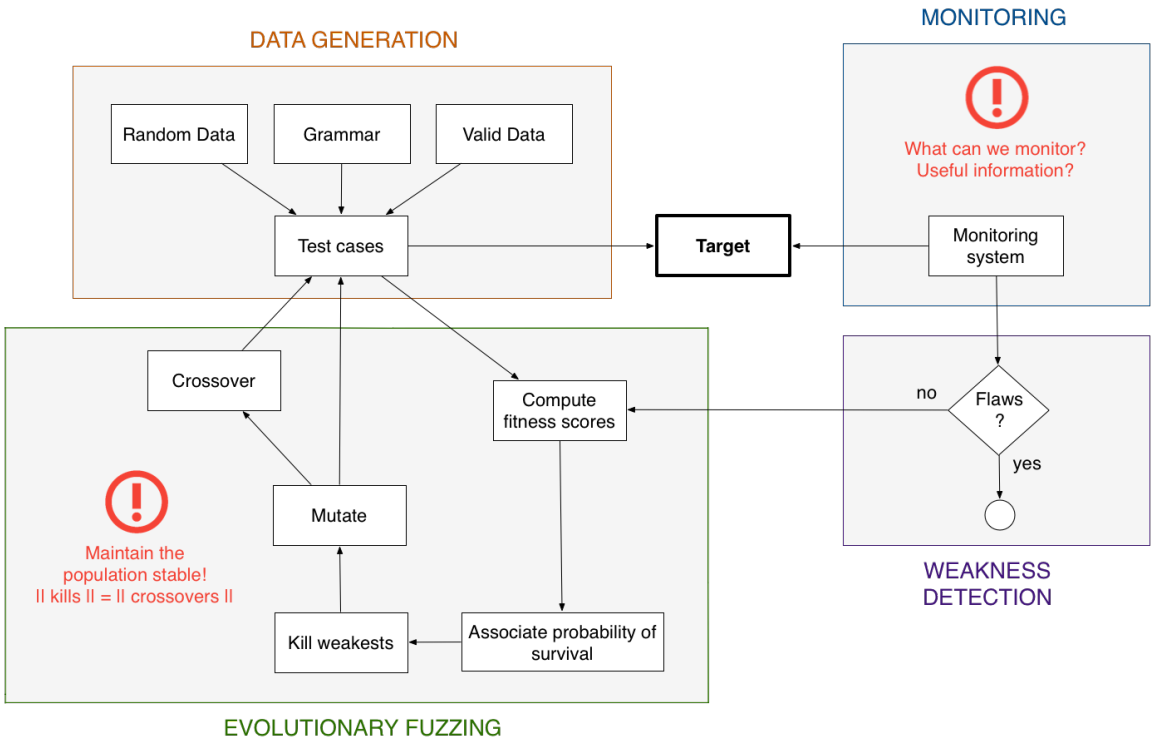


Fig. 1: Evolutionary process

8.2 User interface

An evolutionary process can be configured by extending the *framework.evolutionary_helpers.Population* and *framework.evolutionary_helpers.Individual* abstract classes. These elements describe the contract that needs to be satisfied in order for the evolutionary process to get running. Basically, the methods `_initialize()` and `reset()` can be used to initialize the first population, `evolve()` to get the population to the next generation and `is_final()` to specify a stop criteria.

As these are very generic, they bring a lot of flexibility but require some work. To address this issue, `Fuddly` also proposes a default implementation that describes the classic approach introduced in the previous section. Each step is expressed using one of the *framework.evolutionary_helpers.DefaultPopulation* methods. The evolution stops when the population extincts or if a maximum number of generation exceeds.

- `_compute_scores()`: computes the *individuals* fitness scores, which is, in the default implementation, a random score between 0 and 100. This implementation have to be overridden to match the context. Indeed, this method is used to characterize the *adaptation* of each test case to the target, meaning the negative impact it had on the target. Besides, it also deals with the diversity of the population in order to avoid its premature extinction.
- `_compute_probability_of_survival()`: simply normalize fitness scores between 0 and 1.
- `_kill()`: rolls the dices !
- `_mutate()`: operates three bit flips on each individual using the stateless disruptor `C`.
- `_crossover()`: compensates the kills through the use of a crossover algorithm which is configurable.

Finally, to make an evolutionary process available to the framework, it has to be registered at project level (meaning inside a `*_proj.py` file), through `framework.Project.register_evolutionary_process()`. This method expects processes in the form of 3-tuples containing:

- a name for the scenario that will implement the evolutionary process;
- a class that inherits from *framework.evolutionary_helpers.Population*;
- and parameters that will be passed to the *framework.evolutionary_helpers.EvolutionaryScenariosFactory* in order to instantiate the appropriate population object.

Here under is provided an example to register an evolutionary process (defined in `tuto_proj.py`):

```
from framework.evolutionary_helpers import DefaultPopulation

init_dp1 = DataProcess([('tTYPE', UI(fuzz_mag=0.2))], seed='exist_cond')
init_dp1.append_new_process([('tSTRUCT', UI(deep=True))])

project.register_evolutionary_processes(
    ('evol1',
     DefaultPopulation,
     {'init_process': init_dp1,
      'max_size': 80,
      'max_generation_nb': 3,
      'crossover_algo': CrossoverHelper.crossover_algo1})
)
```

Once loaded from `Fuddly`, `Scenario` are created from registered evolutionary processes, which are callable (like any other scenarios) through their associated `Generator`. In our example, only one process is registered and will lead to the creation of the generator `SC_EVOL1`. After each call to it, the evolutionary process will progress and a new test case will be produced.

Note that the *framework.evolutionary_helpers.DefaultPopulation* is used with this scenario. It expects the following parameters:

- The first one describe the process to follow to generate the data in the initial population (refer to the API documentation for more information). In the example, the process enables to generate altered data from the data type `exist_cond` thanks to the the disruptors `tTYPE` and `tSTRUCT`.
- The second specify the maximum size of the population.
- The third is a criteria to stop the evolutionary process. It provides the maximum number of generation to reach
- The fourth is the crossover algorithm to be used. You can either provide your own implementation or use the ones available in *framework.evolutionary_helpers.CrossoverHelper*. Refer to *Crossover Algorithms* for more information.

8.3 Crossover Algorithms

The evolutionary fuzzing introduces two crossover algorithms that can be used within the crossover operation.

8.3.1 Algo1 - Randomly swap some root nodes' children

Description: Produce two nodes by swapping some of the children of two given graphs roots.

Reference: *framework.evolutionary_helpers.CrossoverHelper.crossover_algo1()*

8.3.2 Algo2 - Randomly swap some leaf nodes

Description: Produce two children by making two graphs swap a given percentages of their leaf nodes.

Reference: *framework.evolutionary_helpers.CrossoverHelper.get_configured_crossover_algo2()*

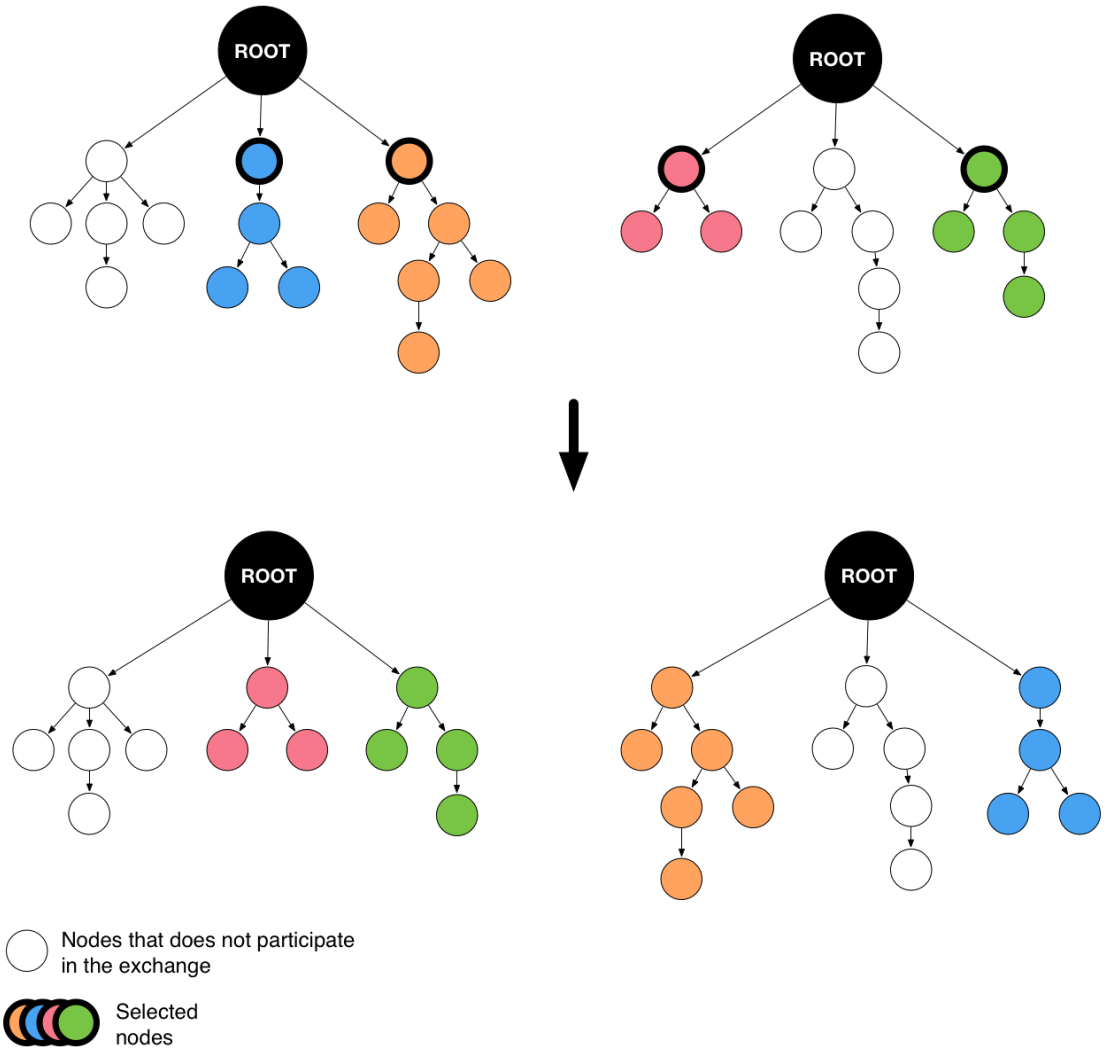


Fig. 2: Algo1 example

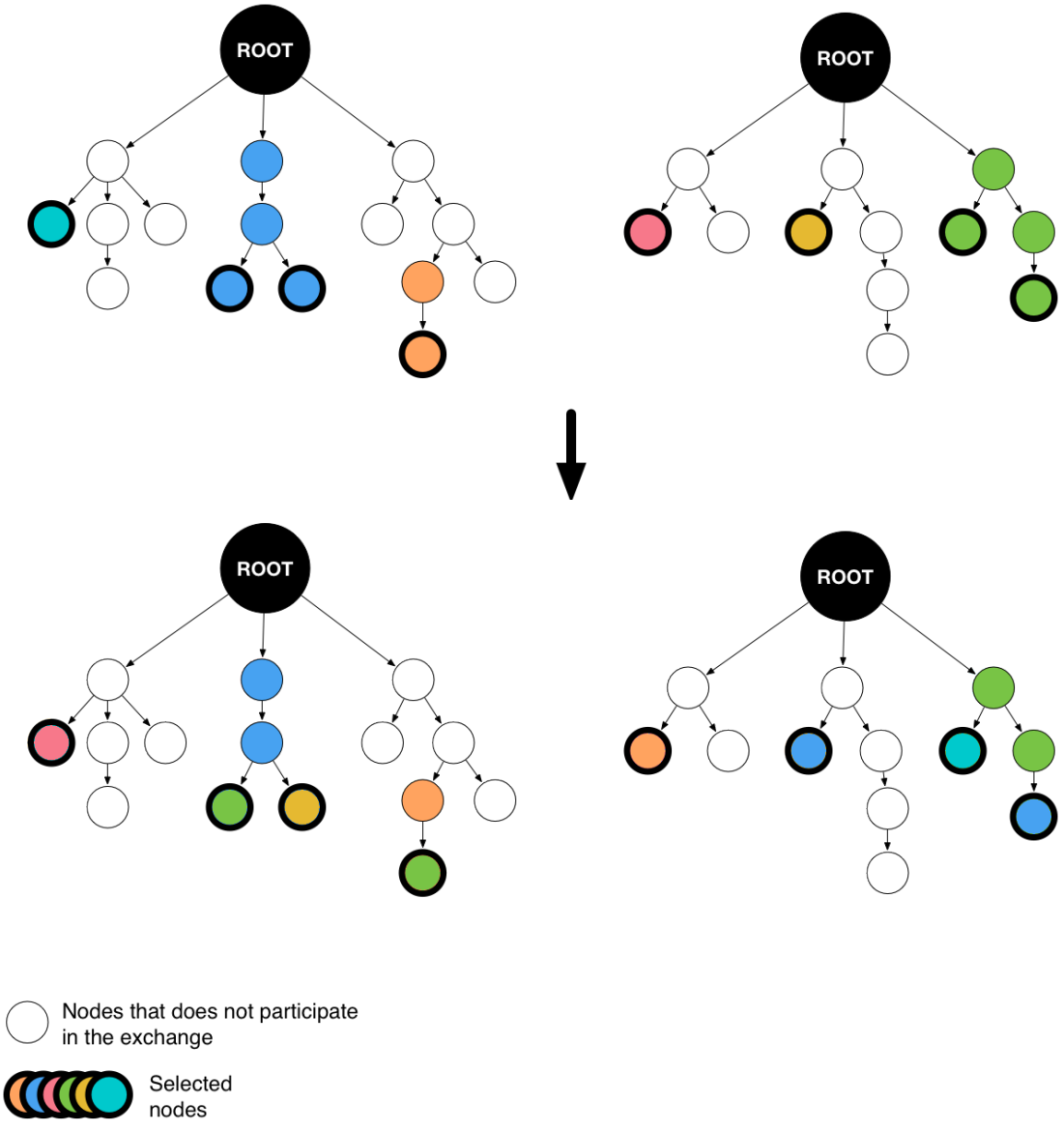


Fig. 3: Algo2 example

CHAPTER

NINE

GENERIC DATA MAKERS

9.1 Generic Generators

The current generic generators are presented within the following sections.

9.1.1 GENP - Pattern Generation

Description: Generate basic data based on a pattern and different parameters.

Reference: *framework.generic_data_makers.g_generic_pattern*

Parameters:

```
|_ pattern
|      | desc: Pattern to be used for generating data
|      | default: b'1234567890' [type: bytes]
|_ prefix
|      | desc: Prefix added to the pattern
|      | default: b'' [type: bytes]
|_ suffix
|      | desc: Suffix replacing the end of the pattern
|      | default: b'' [type: bytes]
|_ size
|      | desc: Size of the generated data.
|      | default: None [type: int]
|_ eval
|      | desc: The pattern will be evaluated before being used. Note that the
|      |       evaluation shall result in a byte string.
|      | default: False [type: bool]
```

9.1.2 POPULATION - Generator for Evolutionary Fuzzing

This generator is used only internally by the evolutionary fuzzing infrastructure.

9.2 Generic Disruptors

The current generic disruptors are presented within the following sections.

9.2.1 Stateful Disruptors

9.2.1.1 tTYPE - Advanced Alteration of Terminal Typed Node

Description: Perform alterations on typed nodes (one at a time) according to:

- their type (e.g., INT, Strings, ...)
- their attributes (e.g., allowed values, minimum size, ...)
- knowledge retrieved from the data (e.g., if the input data uses separators, their symbols are leveraged in the fuzzing)
- knowledge on the target retrieved from the project file or dynamically from feedback inspection (e.g., C language, GNU/Linux OS, ...)

If the input has different shapes (described in non-terminal nodes), this will be taken into account by fuzzing every shape combinations.

Note: this disruptor includes what tSEP does and goes beyond with respect to separators.

Reference: *framework.generic_data_makers.sd_fuzz_typed_nodes*

Parameters:

```
|_ init
|      | desc: make the model walker ignore all the steps until the provided
|      |       one
|      | default: 1 [type: int]
|_ max_steps
|      | desc: maximum number of steps (-1 means until the end)
|      | default: -1 [type: int]
|_ min_node_tc
|      | desc: Minimum number of test cases per node (-1 means until the end)
|      | default: -1 [type: int]
|_ max_node_tc
|      | desc: Maximum number of test cases per node (-1 means until the end).
|      |       This value is used for nodes with a fuzz weight strictly greater
|      |       than 1.
|      | default: -1 [type: int]
|_ clone_node
|      | desc: if True, this operator will always return a copy of the node. (for
|      |       stateless disruptors dealing with big data it can be useful
|      |       to it to False)
|      | default: True [type: bool]
|_ path
|      | desc: Graph path regexp to select nodes on which the disruptor should
|      |       apply
|      | default: None [type: str]
|_ sem
|      | desc: Semantics to select nodes on which the disruptor should apply.
|      | default: None [type: str, list]
```

(continues on next page)

(continued from previous page)

```
|_ deep
|      | desc: When set to True, if a node structure has changed, the modelwalker
|      |       will reset its walk through the children nodes
|      | default: True [type: bool]
|_ full_combinatory
|      | desc: When set to True, enable full-combinatory mode for non-terminal
|      |       nodes. It means that the non-terminal nodes will be customized
|      |       in "FullCombinatory" mode
|      | default: False [type: bool]
|_ ign_sep
|      | desc: when set to True, separators will be ignored if
|      |       any are defined.
|      | default: False [type: bool]
|_ fix
|      | desc: limit constraints fixing to the nodes related to the currently
|      |       fuzzed one (only implemented for 'sync_size_with' and
|      |       'sync_enc_size_with')
|      | default: True [type: bool]
|_ fix_all
|      | desc: for each produced data, reevaluate the constraints on the whole
|      |       graph
|      | default: False [type: bool]
|_ order
|      | desc: when set to True, the fuzzing order is strictly guided by the
|      |       data structure. Otherwise, fuzz weight (if specified in the
|      |       data model) is used for ordering
|      | default: False [type: bool]
|_ fuzz_mag
|      | desc: order of magnitude for maximum size of some fuzzing test cases.
|      | default: 1.0 [type: float]
|_ make_determinist
|      | desc: If set to 'True', the whole model will be set in determinist mode.
|      |       Otherwise it will be guided by the data model determinism.
|      | default: False [type: bool]
|_ leaf_fuzz_determinism
|      | desc: If set to 'True', each typed node will be fuzzed in a deterministic
|      |       way. If set to 'False' each typed node will be fuzzed in a random
|      |       way. Otherwise, if it is set to 'None', it will be guided by
|      |       the data model determinism. Note: this option is complementary
|      |       to 'determinism' as it acts on the typed node substitutions
|      |       that occur through this disruptor
|      | default: True [type: bool]
|_ leaf_determinism
|      | desc: If set to 'True', all the typed nodes of the model will be set
|      |       to determinist mode prior to any fuzzing. If set to 'False',
|      |       they will be set to random mode. Otherwise, if set to 'None',
|      |       nothing will be done.
|      | default: None [type: bool]
|_ consider_sibbling_change
|      | desc: While walking through terminal nodes, if sibbling nodes are
|      |       no more the same because of existence condition for instance,
|      |       walk through the new nodes.
```

(continues on next page)

(continued from previous page)

```
|      | default: True [type: bool]
|_ ign_mutable_attr
|      | desc: Walk through all the nodes even if their Mutable attribute is
|      |       cleared.
|      | default: False [type: bool]
```

9.2.1.2 tSTRUCT - Alter Data Structure

Description: Perform constraints alteration (one at a time) on each node that depends on another one regarding its existence, its quantity, its size, ...

If *deep* is set, enable more corruption cases on the data structure, based on the internals of each non-terminal node:

- the minimum and maximum amount of the subnodes of each non-terminal nodes
- ...

Reference: *framework.generic_data_makers.sd_struct_constraints*

Parameters:

```
|_ init
|      | desc: make the model walker ignore all the steps until the provided
|      |       one
|      | default: 1 [type: int]
|_ max_steps
|      | desc: maximum number of steps (-1 means until the end)
|      | default: -1 [type: int]
|_ path
|      | desc: graph path regexp to select nodes on which the disruptor should
|      |       apply
|      | default: None [type: str]
|_ sem
|      | desc: Semantics to select nodes on which the disruptor should apply.
|      | default: None [type: str, list]
|_ deep
|      | desc: if True, enable corruption of non-terminal node internals
|      | default: False [type: bool]
```

Usage Example: A typical *disruptor chain* for leveraging this disruptor could be:

```
<Data Generator> tWALK(path='path/to/some/node') tSTRUCT
```

Note: Test this chain with the data example found at *How to Describe a Data Format Whose Parts Change Depending on Some Fields*, and set the path to the `opcode` node path.

See also:

Refer to *How to Perform Automatic Modification on Data* for insight into *disruptor chains*.

9.2.1.3 tALT - Walk Through Alternative Node Configurations

Description: Switch the configuration of each node, one by one, with the provided alternate configuration.

Reference: *framework.generic_data_makers.sd_switch_to_alternate_conf*

Parameters:

```
|_ clone_node
|      | desc: if True, this operator will always return a copy of the node. (for
|      |       stateless disruptors dealing with big data it can be useful
|      |       to it to False)
|      | default: True [type: bool]
|_ init
|      | desc: make the model walker ignore all the steps until the provided
|      |       one
|      | default: 1 [type: int]
|_ max_steps
|      | desc: maximum number of steps (-1 means until the end)
|      | default: -1 [type: int]
|_ min_node_tc
|      | desc: Minimum number of test cases per node (-1 means until the end)
|      | default: -1 [type: int]
|_ max_node_tc
|      | desc: Maximum number of test cases per node (-1 means until the end).
|      |       This value is used for nodes with a fuzz weight strictly greater
|      |       than 1.
|      | default: -1 [type: int]
|_ conf
|      | desc: Change the configuration, with the one provided (by name), of
|      |       all nodes reachable from the root, one-by-one. [default value
|      |       is set dynamically with the first-found existing alternate␣
↪configuration]
|      | default: None [type: str, list, tuple]
```

9.2.1.4 tCONST - Alteration of Constraints

Description: When the CSP (Constraint Satisfiability Problem) backend are used in the node description. This operator negates the constraint one-by-one and output 1 or more samples for each negated constraint.

Reference: *framework.generic_data_makers.sd_constraint_fuzz*

Parameters:

```
|_ const_idx
|      | desc: Index of the constraint to begin with (first index is 1)
|      | default: 1 [type: int]
|_ sample_idx
|      | desc: Index of the sample for the selected constraint to begin with
|      |       (first index is 1)
|      | default: 1 [type: int]
|_ clone_node
|      | desc: If True, this operator will always return a copy of the node.
|      |       (for stateless diruptors dealing with big data it can be usefull
```

(continues on next page)

(continued from previous page)

```
|      |       to set it to False)
|      | default: True [type: bool]
|_ samples_per_cst
|      | desc: Maximum number of samples to output for each negated constraint
|      |       (-1 means until the end)
|      | default: -1 [type: int]
```

9.2.1.5 tSEP - Alteration of Separator Node

Description: Perform alterations on separators (one at a time). Each time a separator is encountered in the provided data, it will be replaced by another separator picked from the ones existing within the provided data.

Reference: *framework.generic_data_makers.sd_fuzz_separator_nodes*

Parameters:

```
|_ clone_node
|      | desc: if True, this operator will always return a copy of the node. (for
|      |       stateless disruptors dealing with big data it can be useful
|      |       to it to False)
|      | default: True [type: bool]
|_ init
|      | desc: make the model walker ignore all the steps until the provided
|      |       one
|      | default: 1 [type: int]
|_ max_steps
|      | desc: maximum number of steps (-1 means until the end)
|      | default: -1 [type: int]
|_ min_node_tc
|      | desc: Minimum number of test cases per node (-1 means until the end)
|      | default: -1 [type: int]
|_ max_node_tc
|      | desc: Maximum number of test cases per node (-1 means until the end).
|      |       This value is used for nodes with a fuzz weight strictly greater
|      |       than 1.
|      | default: -1 [type: int]
|_ path
|      | desc: graph path regexp to select nodes on which the disruptor should
|      |       apply
|      | default: None [type: str]
|_ sem
|      | desc: Semantics to select nodes on which the disruptor should apply.
|      | default: None [type: str, list]
|_ order
|      | desc: when set to True, the fuzzing order is strictly guided by the
|      |       data structure. Otherwise, fuzz weight (if specified in the
|      |       data model) is used for ordering
|      | default: False [type: bool]
|_ deep
|      | desc: when set to True, if a node structure has changed, the modelwalker
|      |       will reset its walk through the children nodes
|      | default: True [type: bool]
```

9.2.1.6 tWALK - Walk Through a Data Model

Description: Walk through the provided data and for each visited node, iterates over the allowed values (with respect to the data model). Note: *no alteration* is performed by this disruptor.

Reference: *framework.generic_data_makers.sd_walk_data_model*

Parameters:

```
|_ clone_node
|      | desc: if True, this operator will always return a copy of the node. (for
|      |       stateless disruptors dealing with big data it can be useful
|      |       to it to False)
|      | default: True [type: bool]
|_ init
|      | desc: make the model walker ignore all the steps until the provided
|      |       one
|      | default: 1 [type: int]
|_ max_steps
|      | desc: maximum number of steps (-1 means until the end)
|      | default: -1 [type: int]
|_ min_node_tc
|      | desc: Minimum number of test cases per node (-1 means until the end)
|      | default: -1 [type: int]
|_ max_node_tc
|      | desc: Maximum number of test cases per node (-1 means until the end).
|      |       This value is used for nodes with a fuzz weight strictly greater
|      |       than 1.
|      | default: -1 [type: int]
|_ path
|      | desc: graph path regexp to select nodes on which the disruptor should
|      |       apply
|      | default: None [type: str]
|_ sem
|      | desc: Semantics to select nodes on which the disruptor should apply.
|      | default: None [type: str, list]
|_ full_combinatory
|      | desc: When set to True, enable full-combinatory mode for non-terminal
|      |       nodes. It means that the non-terminal nodes will be customized
|      |       in "FullCombinatory" mode
|      | default: True [type: bool]
|_ leaf_determinism
|      | desc: If set to 'True', all the typed nodes of the model will be set
|      |       to determinist mode prior to any fuzzing. If set to 'False',
|      |       they will be set to random mode. Otherwise, if set to 'None',
|      |       nothing will be done.
|      | default: None [type: bool]
|_ order
|      | desc: when set to True, the walking order is strictly guided by the
|      |       data structure. Otherwise, fuzz weight (if specified in the
|      |       data model) is used for ordering
|      | default: True [type: bool]
|_ nt_only
|      | desc: walk through non-terminal nodes only
```

(continues on next page)

(continued from previous page)

```
|      | default: False [type: bool]
|_ deep
|      | desc: when set to True, if a node structure has changed, the modelwalker
|      |       will reset its walk through the children nodes
|      | default: True [type: bool]
|_ fix_all
|      | desc: for each produced data, reevaluate the constraints on the whole
|      |       graph
|      | default: True [type: bool]
|_ ign_mutable_attr
|      | desc: Walk through all the nodes even if their Mutable attribute is
|      |       cleared.
|      | default: True [type: bool]
```

9.2.1.7 tWALKcsp - Walk Through the Constraint of a Data Model

Description: When the CSP (Constraint Satisfiability Problem) backend are used in the data description. This operator walk through the solutions of the CSP.

Reference: *framework.generic_data_makers.sd_walk_csp_solutions*

Parameters:

```
|_ init
|      | desc: Make the operator ignore all the steps until the provided one
|      | default: 1 [type: int]
|_ clone_node
|      | desc: If True, this operator will always return a copy of the node.
|      |       (for stateless diruptors dealing with big data it can be usefull
|      |       to set it to False)
|      | default: True [type: bool]
|_ notify_exhaustion
|      | desc: When all the solutions of the CSP have been walked through,
|      |       the disruptor will notify it if this parameter is set to True.
|      | default: True [type: bool]
```

9.2.2 Stateless Disruptors

9.2.2.1 ADD - Add Data Within a Node

Description: Add some data within the retrieved input.

Reference: *framework.generic_data_makers.d_add_data*

Parameters:

```
|_ path
|      | desc: Graph path to select the node on which the disruptor should
|      |       apply.
|      | default: None [type: str]
|_ after
|      | desc: If True, the addition will be done after the selected node.
```

(continues on next page)

(continued from previous page)

```
|      |       Otherwise, it will be done before.
|      | default: True [type: bool]
|_ atom
|      | desc: Name of the atom to add within the retrieved input. It is mutually
|      |       exclusive with @raw
|      | default: None [type: str]
|_ raw
|      | desc: Raw value to add within the retrieved input. It is mutually
|      |       exclusive with @atom.
|      | default: b'' [type: bytes, str]
|_ name
|      | desc: If provided, the added node will have this name.
|      | default: None [type: str]
```

9.2.2.2 OP - Perform Operations on Nodes

Description: Perform an operation on the nodes specified by the regexp path. @op is an operation that applies to a node and @params are a tuple containing the parameters that will be provided to @op. If no path is provided, the root node will be used.

Reference: *framework.generic_data_makers.d_operate_on_nodes*

Parameters:

```
|_ path
|      | desc: Graph path regexp to select nodes on which the disruptor should
|      |       apply.
|      | default: None [type: str]
|_ op
|      | desc: The operation to perform on the selected nodes.
|      | default: <function Node.clear_attr> [type: method, function]
|_ op_ref
|      | desc: Predefined operation that can be referenced by name. The current
|      |       predefined function are: 'unfreeze', 'freeze', 'walk'. Take
|      |       precedence over @op if not None.
|      | default: None [type: str]
|_ params
|      | desc: Tuple of parameters that will be provided to the operation.
|      | default: () [type: tuple]
|_ clone_node
|      | desc: If True the dmaker will always return a copy of the node. (For
|      |       stateless disruptors dealing with big data it can be useful
|      |       to set it to False.)
|      | default: False [type: bool]
```

9.2.2.3 MOD - Modify Node Contents

Description: Perform modifications on the provided data. Two ways are possible:

- Either the change is performed on the content of the nodes specified by the *path* parameter with the new *value* provided, and the optional constraints for the absorption (use *node absorption* infrastructure);
- Or the changed is performed based on a dictionary provided through the parameter *multi_mod*

Reference: *framework.generic_data_makers.d_modify_nodes*

Parameters:

```
|_ path
|      | desc: Graph path regexp to select nodes on which the disruptor should
|      |       apply.
|      | default: None [type: str]
|_ sem
|      | desc: Semantics to select nodes on which the disruptor should apply.
|      | default: None [type: str, list]
|_ value
|      | desc: The new value to inject within the data.
|      | default: b'' [type: bytes]
|_ constraints
|      | desc: Constraints for the absorption of the new value.
|      | default: AbsNoCsts() [type: AbsCsts]
|_ multi_mod
|      | desc: Dictionary of <path>:<item> pairs or <NodeSemanticsCriteria>:<item>
|      |       pairs or <NodeInternalsCriteria>:<item> pairs to change multiple
|      |       nodes with different values. <item> can be either only the new
|      |       <value> or a tuple (<value>,<abscsts>) if new constraint for
|      |       absorption is needed
|      | default: None [type: dict]
|_ unfold
|      | desc: Resolve all the generator nodes within the input before performing
|      |       the @path/@sem research
|      | default: False [type: bool]
|_ clone_node
|      | desc: If True the dmaker will always return a copy of the node. (For
|      |       stateless disruptors dealing with big data it can be useful
|      |       to set it to False.)
|      | default: False [type: bool]
```

9.2.2.4 CALL - Call Function

Description: Call the function provided with the first parameter being the *framework.data.Data* object received as input of this disruptor, and optionally with additional parameters if *params* is set. The function should return a *framework.data.Data* object.

The signature of the function should be compatible with:

```
func(data, *args) --> Data()
```

Reference: *framework.generic_data_makers.d_modify_nodes*

Parameters:

```
|_ func
|      | desc: The function that will be called with a node as its first parameter,
|      |       and provided optionnaly with addtionnal parameters if @params
|      |       is set.
|      | default: lambda x: x [type: method, function]
|_ params
|      | desc: Tuple of parameters that will be provided to the function.
|      | default: None [type: tuple]
```

9.2.2.5 NEXT - Next Node Content

Description: Move to the next content of the nodes from input data or from only a piece of it (if the parameter *path* is provided). Basically, unfreeze the nodes then freeze them again, which will consequently produce a new data.

Reference: *framework.generic_data_makers.d_next_node_content*

Parameters:

```
|_ path
|      | desc: graph path regexp to select nodes on which the disruptor should
|      |       apply
|      | default: None [type: str]
|_ clone_node
|      | desc: if True, this operator will always return a copy of the node. (for
|      |       stateless disruptors dealing with big data it can be useful
|      |       to it to False)
|      | default: False [type: bool]
|_ recursive
|      | desc: apply the disruptor recursively
|      | default: True [type: str]
```

9.2.2.6 FIX - Fix Data Constraints

Description: Release constraints from input data or from only a piece of it (if the parameter *path* is provided), then recompute them. By constraints we mean every generator (or function) nodes that may embeds constraints between nodes, and every node *existence conditions*.

See also:

Refer to *How to Describe a Data Format Whose Parts Change Depending on Some Fields* for insight into existence conditions.

Reference: *framework.generic_data_makers.d_fix_constraints*

Parameters:

```
|_ path
|      | desc: graph path regexp to select nodes on which the disruptor should
|      |       apply
|      | default: None [type: str]
|_ clone_node
|      | desc: if True, this operator will always return a copy of the node. (for
|      |       stateless disruptors dealing with big data it can be useful
```

(continues on next page)

(continued from previous page)

```
|      |       to it to False)
|      | default: False [type: bool]
```

9.2.2.7 ALT - Alternative Node Configuration

Description: Switch to an alternate configuration.

Reference: *framework.generic_data_makers.d_switch_to_alternate_conf*

Parameters:

```
|_ path
|      | desc: graph path regexp to select nodes on which the disruptor should
|      |       apply
|      | default: None [type: str]
|_ recursive
|      | desc: does the reachable nodes from the selected ones need also to
|      |       be changed?
|      | default: True [type: bool]
|_ conf
|      | desc: change the configuration, with the one provided (by name), of
|      |       all subnodes fetched by @path, one-by-one. [default value is
|      |       set dynamically with the first-found existing alternate␣
↪configuration]
|      | default: None [type: str]
```

9.2.2.8 C - Node Corruption

Description: Corrupt bits on some nodes of the data model.

Reference: *framework.generic_data_makers.d_corrupt_node_bits*

Parameters:

```
|_ path
|      | desc: graph path regexp to select nodes on which the disruptor should
|      |       apply
|      | default: None [type: str]
|_ nb
|      | desc: apply corruption on @nb Nodes fetched randomly within the data
|      |       model
|      | default: 2 [type: int]
|_ ascii
|      | desc: enforce all outputs to be ascii 7bits
|      | default: False [type: bool]
|_ new_val
|      | desc: if provided change the selected byte with the new one
|      | default: None [type: str]
```

9.2.2.9 Cp - Corruption at Specific Position

Description: Corrupt bit at a specific byte.

Reference: *framework.generic_data_makers.d_corrupt_bits_by_position*

Parameters:

```
|_ new_val
|      | desc: if provided change the selected byte with the new one
|      | default: None [type: str]
|_ ascii
|      | desc: enforce all outputs to be ascii 7bits
|      | default: False [type: bool]
|_ idx
|      | desc: byte index to be corrupted (from 1 to data length)
|      | default: 1 [type: int]
```

9.2.2.10 EXT - Make Use of an External Program

Description: Call an external program to deal with the data.

Reference: *framework.generic_data_makers.d_call_external_program*

Parameters:

```
|_ path
|      | desc: graph path regexp to select nodes on which the disruptor should
|      |       apply
|      | default: None [type: str]
|_ cmd
|      | desc: the command
|      | default: None [type: list, tuple, str]
|_ file_mode
|      | desc: if True the data will be provided through a file to the external
|      |       program, otherwise it will be provided on the command line directly
|      | default: True [type: bool]
```

9.2.2.11 SIZE - Truncate

Description: Truncate the data (or part of the data) to the provided size.

Reference: *framework.generic_data_makers.d_max_size*

Parameters:

```
|_ sz
|      | desc: truncate the data (or part of the data) to the provided size
|      | default: 10 [type: int]
|_ path
|      | desc: graph path regexp to select nodes on which the disruptor should
|      |       apply
|      | default: None [type: str]
```

9.2.2.12 STRUCT - Shake Up Data Structure

Description: Disrupt the data model structure (replace ordered sections by unordered ones).

Reference: *framework.generic_data_makers.d_fuzz_model_structure*

Parameters:

```
|_ path
|      | desc: graph path regexp to select nodes on which the disruptor should
|      |       apply
|      | default: None [type: str]
```

9.2.2.13 COPY - Shallow Copy Data

Description: Shallow copy of the input data, which means: ignore its frozen state during the copy.

Reference: *framework.generic_data_makers.d_shallow_copy*

Note: Random seeds are generally set while loading the data model. This disruptor enables you to reset the seeds for the input data.

CHAPTER

TEN

GENERIC TARGETS

The following section present some generic targets that inherit from *framework.target_helpers.Target*. They can be directly used as is, within your project files (refer to *Defining a Project Environment*), or for some of them they can also be customized by inheriting from them and implementing some intended methods acting as hooks within the generic targets.

Some of them will automatically provide feedback if an error occurs, to make `fuddly` aware of it and act accordingly (refer to *Defining Probes* for more information on that topic).

Additionally, if the generic target support feedback retrieval, the way it is retrieved is guided by a feedback timeout and one of the following mode:

- *framework.target_helpers.Target.FBK_WAIT_FULL_TIME*: Wait for the full time slot allocated for feedback retrieval
- *framework.target_helpers.Target.FBK_WAIT_UNTIL_RECV*: Wait until the target has sent something back to us

The feedback timeout is set through *framework.target_helpers.Target.set_feedback_timeout()*, while the modes are set through *framework.target_helpers.Target.set_feedback_mode()*.

Note: Depending on the generic target, all the feedback modes are not supported.

10.1 NetworkTarget

Reference: *framework.targets.network.NetworkTarget*

Description: This generic target enables you to interact with a network target in TCP or UDP, through one or more interfaces. Each declared interface is customizable, and the generic target itself can be more customized by inheriting from it. Especially, the following methods are expected to be overloaded, depending on the user needs:

- *framework.targets.network.NetworkTarget._custom_data_handling_before_emission()* for performing some actions related to the data that will be emitted right after.
- *framework.targets.network.NetworkTarget._feedback_handling()* for filtering/handling feedback in some ways before transferring it to `fuddly`.
- *framework.targets.network.NetworkTarget.initialize()* for doing specific actions at target initialization.
- *framework.targets.network.NetworkTarget.terminate()* for doing specific actions at target termination.

See also:

Refer also to the tutorial section *Defining the Targets* that guides you through an example of network target.

Feedback: This target will automatically provide feedback on any network-related error encountered while delivering data to the target.

Supported Feedback Mode:

- *framework.target_helpers.Target.FBK_WAIT_FULL_TIME*
- *framework.target_helpers.Target.FBK_WAIT_UNTIL_RECV*

Usage Example:

```
1 tg = NetworkTarget(host='localhost', port=12345, data_semantics='TG1',
2                    hold_connection=True)
3 tg.register_new_interface(host='localhost', port=54321,
4                            socket_type=(socket.AF_INET, socket.SOCK_STREAM),
5                            data_semantics='TG2', server_mode=True, hold_
  ↪connection=True)
6 tg.add_additional_feedback_interface('localhost', 7777,
7                                       socket_type=(socket.AF_INET, socket.SOCK_
  ↪DGRAM),
8                                       fbk_id='My Feedback Source', server_mode=True)
9 tg.set_timeout(fbk_timeout=5, sending_delay=3)
```

line 1-2 We instantiate the `NetworkTarget` by providing the parameters of the first interface: a TCP connection to `localhost` on port `12345`. We specify that the connection on this interface have to be maintained between each data emission to the target though the parameter `hold_connection`. This interface will be considered as the default one, through which any data will be routed to unless specified differently, through the `data_semantics` parameter (look at the class API for insight). In this case, data without *semantic*, and data with a *semantic* equal to `'TG1'` will go through this interface.

line 3-5 We declare another interface where we specify the real target will connect to us (and not otherwise), by using the `server_mode` parameter. We also set semantics to `'TG2'` which means that only data marked with such semantics will be routed to this interface.

line 6-8 We declare another interface for only feedback purpose, where the source of the feedback will send data to us in UDP (`socket.SOCK_DGRAM`) on the port `7777`. Note that an identifier has to be provided (`fbk_id`), and will be used to refer to the interface at different points in time. Main interfaces (the first one and the ones defined through *framework.targets.network.NetworkTarget.register_new_interface()*) has also an identifier but it is set automatically by the `NetworkTarget`.

line 9 We set some time constraints: `fbk_timeout` for gathering feedback from all the interfaces; `sending_delay` for sending data to the target (client mode) or waiting for client connections before sending data to them (server mode). Note this method is specific to this target and remains consistent with *framework.target_helpers.Target.set_feedback_timeout()*.

10.2 LocalTarget

Reference: *framework.targets.local.LocalTarget*

Description: This generic target enables you to interact with a program running on the same platform as `fuddly`. It can be customized by inheriting from it. The following methods are expected to be overloaded, depending on the user needs:

- *framework.targets.local.LocalTarget.initialize()* for doing specific actions at target initialization.
- *framework.targets.local.LocalTarget.terminate()* for doing specific actions at target termination.

Feedback: This target will automatically provide feedback if the application writes on `stderr` or returns a negative status or terminates/crashes. `stdout` can also be parsed looking for user-provided keywords that will trigger some feedback with negative status or even parsed by a user-provided function.

Supported Feedback Mode:

- *framework.target_helpers.Target.FBK_WAIT_UNTIL_RECV*

Usage example:

```
1 import framework.global_resources as gr
2
3 tg = LocalTarget(tmpfile_ext='.zip')
4 tg.set_target_path('unzip')
5 tg.set_post_args('-d ' + gr.workspace_folder)
```

line 3 We declare a `LocalTarget` and specify the file extension that will be used for interacting with the targeted program.

line 4 We set the file system path to the targeted program.

line 5 We set some parameters that will be used by `fuddly` to make up the command to execute for interacting with the targeted program. This parameter will be put after the file name, but you can also add parameters before it through the method *framework.targets.local.LocalTarget.set_pre_args()*. Note the use of the variable `workspace_folder` that points to the `fuddly` workspace directory which is typically used when temporary files need to be created.

10.3 SSHTarget

Reference: *framework.targets.ssh.SSHTarget*

Description: This generic target enables you to interact with a remote target requiring an SSH connection.

Feedback: This target will automatically provide the results of the commands sent through SSH.

Supported Feedback Mode:

- *framework.target_helpers.Target.FBK_WAIT_FULL_TIME*
- *framework.target_helpers.Target.FBK_WAIT_UNTIL_RECV*

Usage Example:

```
1 tg = SSHTarget(host='192.168.0.1', port=22, username='test', password='test')
```

10.4 PrinterTarget

Reference: *framework.targets.printer.PrinterTarget*

Description: This generic target enables you to interact with a IPP server.

Feedback: No feedback is automatically returned.

Usage Example:

```
1 tg = PrinterTarget(tmpfile_ext='.png')
2 tg.set_target_ip('127.0.0.1')
3 tg.set_target_port(631)     # optional
4 tg.set_printer_name('PDF')  # optional
```

line 1 We declare a `PrinterTarget` and specify the file extension that will be used for interacting with the targeted program.

line 2 We set the IP of the IPP server managing the printer.

line 3 We set the port for communicating with the printer.

line 4 We set the name of the printer of interest.

10.5 SIMTarget

Reference: `framework.targets.sim.SIMTarget`

Description: This generic target enables you to interact with a SIM card through a serial line (e.g., a SIM card embedded within an USB GSM modem)

Feedback: This target will automatically provide feedback if an error is received through the serial line used to interact with the SIM card.

Supported Feedback Mode:

- *framework.target_helpers.Target.FBK_WAIT_FULL_TIME*

Usage Example:

```
1 tg = SIMTarget(serial_port='/dev/ttyUSB3', baudrate=115200, pin_code='0000'
2                targeted_tel_num='0123456789', zone='33')
```

10.6 TestTarget

Reference: *framework.targets.debug.TestTarget*

Description: This generic target enables you to stimulate a virtual target that could be useful for test preparation for instance. Some parameters enable to change the behavior of this target.

Feedback: This target could provide random feedback, or feedback chosen from a provided sample list, or it could repeat the received data as its feedback.

Supported Feedback Mode:

- *framework.target_helpers.Target.FBK_WAIT_FULL_TIME*
- *framework.target_helpers.Target.FBK_WAIT_UNTIL_RECV*

Usage Example:

```
tg = TestTarget(name='mytest_target', fbk_samples=['OK','ERROR'])
```

CHAPTER

ELEVEN

GENERIC PROBES AND BACKEND

The following section present some generic probes that inherit from *framework.monitor.Probe*. They can be used within your project files (refer to *Defining a Project Environment*) by inheriting from them and providing the expected parameters. Besides, you have to provide them with a means to access the monitored system, namely a `framework.monitor.Backend`. Note that you can use the same backend for simultaneous probes.

See also:

To define your own probe refer to *Defining Probes*.

Let's illustrate this with the following example where two probes are used to monitor a process through an SSH connection. One is used to check if the PID of the process has changed after each data sending, and the other one to check if the memory used by the process has exceeded its initial memory footprint by 5% (with a probing period of 0.2 second).

The project file should look like this:

```
# Assuming your Project() is referred by the 'project' variable

ssh_backend = SSH_Backend(username='user', password='pass',
                          sshd_ip='127.0.0.1', sshd_port=22)

@blocking_probe(project)
class health_check(ProbePID):
    process_name = 'the_process_to_monitor'
    backend = ssh_backend

@probe(project)
class probe_mem(ProbeMem):
    process_name = 'the_process_to_monitor'
    tolerance = 5
    backend = ssh_backend

targets = [ (YourTarget(), probe_pid, (probe_mem, 0.2)) ]
```

11.1 Generic Backend

See also:

Refer to the class documentation for more details.

11.1.1 SSH_Backend

Reference: *framework.comm_backends.SSH_Backend*

Description: This generic backend enables you to interact with a monitored system through an SSH connection.

11.1.2 Serial_Backend

Reference: *framework.comm_backends.Serial_Backend*

Description: This generic backend enables you to interact with a monitored system through an serial line.

11.1.3 Shell_Backend

Reference: *framework.comm_backends.Shell_Backend*

Description: This generic backend enables you to interact with a local monitored system through a shell.

11.2 Generic Probes

See also:

Refer to the class documentation for more details.

11.2.1 ProbePID

Reference: *framework.monitor.ProbePID*

Description: This generic probe enables you to monitor any modification of a process PID, by specifying its name through the parameter `process_name`.

11.2.2 ProbeMem

Reference: *framework.monitor.ProbeMem*

Description: Generic probe that enables you to monitor the process memory (RSS...) consumption. It can be done by specifying a `threshold` and/or a `tolerance` ratio.

11.2.3 ProbeCmd

Reference: *framework.monitor.ProbeCmd*

Description: Generic probe that enables you to execute shell commands and retrieve the output.

CHAPTER

TWELVE

USEFUL EXAMPLES

12.1 ZIP archive modification

The following example (refer to the figure *below*) illustrates how to modify the second file contents of a ZIP archive, and let `fuddly` recalculate every constraints for you.

```
abszip = dm.get_atom('ZIP')
abszip.set_current_conf('ABS', recursive=True)
abszip.absorb(zip_buff, constraints=AbsNoCsts(size=True,struct=True)

abszip['ZIP/file_list/file:2/data'][0].absorb(b'TEST', constraints=AbsNoCsts())
abszip.unfreeze(only_generators=True)
abszip.get_value()
```

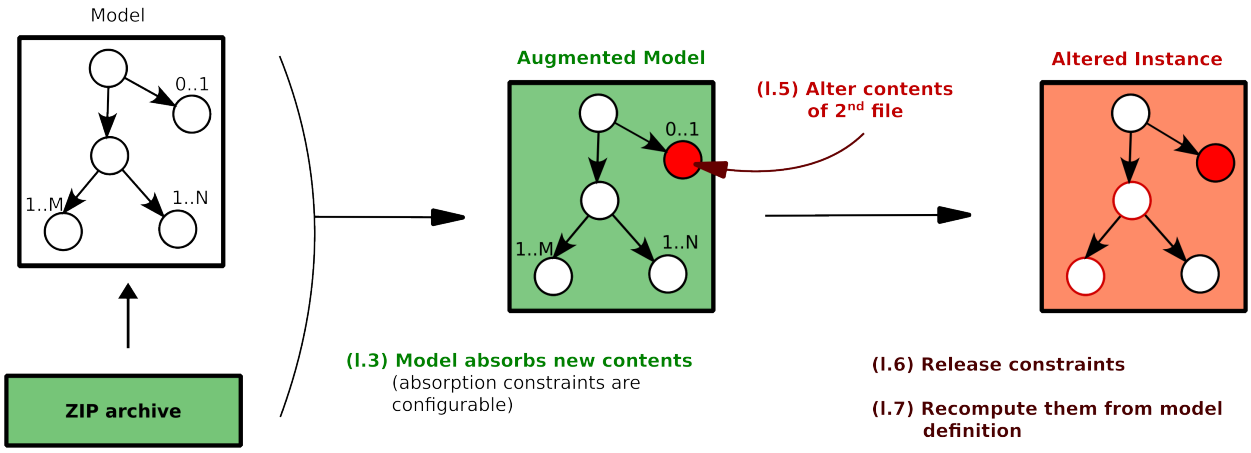


Fig. 1: ZIP archive second file contents modification

CHAPTER

THIRTEEN

FUDDLY API

13.1 API Index

- genindex
- modindex
- search

13.2 framework package

13.2.1 framework.basic_primitives module

framework.basic_primitives.**calc_parity_bit**(*x*)
return 0 if the number of bits is even, otherwise returns 1

framework.basic_primitives.**corrupt_bits**(*s*, *p=0.01*, *n=None*, *ascii=False*)
Flip a given percentage or number of bits from a string

framework.basic_primitives.**corrupt_bytes**(*s*, *p=0.01*, *n=None*, *ctrl_char=False*)
Corrupt a given percentage or number of bytes from a string

framework.basic_primitives.**rand_string**(*size=None*, *min=1*, *max=10*, *str_set='0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUV ./:;<=>?@[\\]^_`{|}~ \t\n\r\x0b\x0c'*)

13.2.2 framework.data module

class framework.data.**AttrGroup**(*attrs_desc*)
Bases: object

__copy__()

__init__(*attrs_desc*)

__module__ = 'framework.data'

clear(*name*)

copy_from(*attr_group*)

is_set(*name*)

set(*name*)

class framework.data.**CallBackOps**(*remove_cb=False*, *stop_process_cb=False*, *ignore_no_data=False*)
Bases: object

Add_PeriodicData = 10

Del_PeriodicData = 11

ForceDataHandling = 3

RemoveCB = 1

Replace_Data = 30

Set_FbkTimeout = 21

Start_Task = 12

StopProcessingCB = 2

Stop_Task = 13

__init__(*remove_cb=False*, *stop_process_cb=False*, *ignore_no_data=False*)

__module__ = 'framework.data'

add_operation(*instr_type*, *id=None*, *param=None*, *period=None*)

get_operations()

is_flag_set(*name*)

set_flag(*name*)

class framework.data.**Data**(*content=None*, *altered=False*, *tg_ids=None*, *description=None*)
Bases: object

__copy__()

__init__(*content=None*, *altered=False*, *tg_ids=None*, *description=None*)

__module__ = 'framework.data'

__repr__()
Return repr(self).

__str__()
Return str(self).

_empty_data_backend = <framework.data.EmptyBackend object>

add_info(*info_str*)

bind_info(*dmaker_type*, *data_maker_name*)

cleanup_all_callbacks()

cleanup_callbacks(*hook=HOOK.after_fbk*)

property content

copy_callback_from(*data*)

generate_info_from_content(*data=None*, *origin=None*, *additional_info=None*)

get_content(*do_copy=False*)

get_data_id()

get_data_model()

get_history()

get_initial_dmaker()

get_length()

has_info()

has_node_content()

has_raw_content()

info_exists(*dmaker_type*, *data_maker_name*, *info*)

is_blocked()

is_empty()

is_recordable()

is_unusable()

make_blocked()

make_free()

make_recordable()

make_unusable()

property origin

pending_callback_ops(*hook=HOOK.after_fbk*)

pretty_print(*raw_limit=200*, *log_func=<built-in method write of _io.TextIOWrapper object>*)

read_info(*dmaker_type*, *data_maker_name*)

register_callback(*callback*, *hook=HOOK.after_fbk*)

remove_info_from(*dmaker_type*, *data_maker_name*)

reset_history()

run_callbacks(*feedback=None*, *hook=HOOK.after_fbk*)

set_attributes_from(*attr_group*)

set_basic_attributes(*from_data=None*)

set_data_id(*data_id*)

set_data_model(*dm*)

set_history(*hist*)

set_initial_dmaker(*t*)

show(*raw_limit=200*, *log_func=<built-in method write of _io.TextIOWrapper object>*)

take_info_ownership(*keep_previous_info=True*)

property tg_ids

to_bytes()

to_formatted_str()

to_str()

update_from(*obj*)

class framework.data.**DataAttr**(*attrs_to_set=None*, *attrs_to_clear=None*)
Bases: *framework.data.AttrGroup*

Reset_DMakers = 1

__annotations__ = {}

__init__(*attrs_to_set=None*, *attrs_to_clear=None*)

__module__ = 'framework.data'

class framework.data.**DataBackend**(*content=None*)
Bases: object

__init__(*content=None*)

__module__ = 'framework.data'

property content

property data_maker_name

property data_maker_type

property data_model

get_content(*do_copy=False*, *materialize=True*)

get_length()

show(*raw_limit=200*, *log_func=<built-in method write of _io.TextIOWrapper object>*)

to_bytes()

to_str()

update_from(*obj*)

class framework.data.**DataProcess**(*process*, *seed=None*, *tg_ids=None*, *auto_regen=False*)
Bases: object

__copy__()

__init__(*process*, *seed=None*, *tg_ids=None*, *auto_regen=False*)
Describe a process to generate a data.

Parameters

- **process** (*list*) – List of disruptors (possibly complemented by parameters) to apply to a seed. However, if the list begin with a generator, the disruptor chain will apply to the outcome of the generator. The generic form for a process is: [action_1, (action_2, UI_2), ... action_n] where action_N can be either: dmaker_type_N or (dmaker_type_N, dmaker_name_N)
- **seed** – (Optional) Can be a registered framework.data_model.Node name or a framework.data_model.Data. Will be provided to the first disruptor in the disruptor chain (described by the parameter process) if it does not begin with a generator.
- **tg_ids** (*list*) – Virtual ID list of the targets to which the outcomes of this data process will be sent. If None, the outcomes will be sent to the first target that has been enabled. In the context of scenario, it embeds virtual IDs.
- **auto_regen** (*boolean*) – If True, the data process will be in a state requesting the framework to rerun the data maker chain after a disruptor yielded (meaning it is exhausted with the data provided to it). It will make the chain going on with new data coming either from the first non-exhausted disruptor (preceding the exhausted one), or from the generator if all

disruptors are exhausted. If `False`, the data process won't be in this state and the framework won't rerun the data maker chain once a disruptor yield. It means the framework will alert about this issue to the end-user, or when used within a Scenario, it will redirect the decision to the scenario itself (this condition may trigger a transition in the scenario).

__module__ = 'framework.data'

__repr__()
Return repr(self).

append_new_process(*process*)
Append a new process to the list.

formatted_str(*oneliner=False*)

make_blocked()

make_free()

next_process()

property process

property process_qty

reset()

class framework.data.**EmptyBackend**(*content=None*)
Bases: *framework.data.DataBackend*

__annotations__ = {}

__module__ = 'framework.data'

property content

property data_maker_name

property data_maker_type

property data_model

get_content(*do_copy=False*, *materialize=True*)

get_length()

to_bytes()

to_formatted_str()

to_str()

class framework.data.**EmptyDataProcess**(*seed=None*, *tg_ids=None*, *auto_regen=False*)
Bases: `object`

__init__(*seed=None*, *tg_ids=None*, *auto_regen=False*)

__module__ = 'framework.data'

class framework.data.**NodeBackend**(*content=None*)
Bases: *framework.data.DataBackend*

__annotations__ = {}

__copy__()

__module__ = 'framework.data'

property content

property data_maker_name

property data_maker_type

get_content(*do_copy=False*, *materialize=True*)

get_length()

show(*raw_limit=200*, *log_func=<built-in method write of _io.TextIOWrapper object>*)

to_bytes()

to_formatted_str()

to_str()

update_from(*obj:* framework.node.Node)

class framework.data.RawBackend(*content=None*)

Bases: *framework.data.DataBackend*

__annotations__ = {}

__module__ = 'framework.data'

property content

property data_maker_name

property data_maker_type

get_content(*do_copy=False*, *materialize=True*)

get_length()

show(*raw_limit=200*, *log_func=<built-in method write of _io.TextIOWrapper object>*)

to_bytes()

to_formatted_str()

to_str()

update_from(*obj*)

13.2.3 framework.data_model module

class framework.data_model.DataModel

Bases: object

Data Model Abstraction

__init__()

__module__ = 'framework.data_model'

__str__()

Return str(self).

_atom_absorption_additional_actions(*atom*)

Called by .create_atom_from_raw_data(). Should be overloaded if specific actions need to be performed on the atoms created from imported raw data

Parameters **atom** – Atom that are going to be registered after being absorbed from raw data

Returns An atom and a short description of the actions

_backend(*atom*)

_create_atom_from_raw_data_specific(*data*, *idx*, *filename*)

Overload this method when creating a node from binary strings need more actions than performing a node absorption.

Parameters

- **data** (*bytes*) – file content
- **idx** (*int*) – index of the imported file
- **filename** (*str*) – name of the imported file

Returns An atom or None

absorb(*data*, *scope=None*, *atom_name=None*, *requested_abs_csts=None*)

Parameters

- **data** –
- **atom_name** (*str*) – requested atom name for the decoding (linked to self.register_atom_for_decoding)
- **scope** (*str*) – requested scope for the decoding (linked to self.register_atom_for_decoding)
- **requested_abs_csts** –

Returns Node which is the result of the absorption or None

Return type *Node*

atom_identifiers()

build_data_model()

This method is called when a data model is loaded. It is called only the first time the data model is loaded. To be implemented by the user.

cleanup()

create_atom_from_raw_data(*data*, *idx*, *filename*)

This function is called for each files (with the right extension) present in imported_data/<data_model_name> and absorb their content by leveraging the atoms of the data model registered for absorption or if none are registered, either call the method _create_atom_from_raw_data_specific() if it is defined or wrap their content in a *framework.node.Node*.

Parameters

- **data** (*bytes*) – file content
- **idx** (*int*) – index of the imported file
- **filename** (*str*) – name of the imported file

Returns An atom or None

customize_node_backend(*default_gen_custo=None*, *default_nonterm_custo=None*)

decode(*data*, *scope=None*, *atom_name=None*, *requested_abs_csts=None*, *colorized=True*)

Parameters

- **data** –

- **atom_name** (*str*) – requested atom name for the decoding (linked to self.register_atom_for_decoding)
- **scope** (*str*) – requested scope for the decoding (linked to self.register_atom_for_decoding)
- **requested_abs_csts** –
- **colorized** –

Returns Node which is the result of the absorption or None and Textual description of the result

Return type tuple

file_extension = 'bin'

get_atom(*hash_key*, *name=None*)

get_atom_for_absorption(*hash_key*)

get_external_atom(*dm_name*, *data_id*, *name=None*)

get_import_directory_path(*subdir=None*)

import_file_contents(*extension=None*, *absorber=None*, *subdir=None*, *path=None*, *filename=None*)

property included_models

knowledge_source = None

load_data_model(*dm_db*)

merge_with(*data_model*)

name = None

pre_build()
This method is called when a data model is loaded. It is executed before build_data_model(). To be implemented by the user.

register(**atom_list*)

register_atom_for_decoding(*atom*, *absorb_constraints=AbsFullCsts()*, *decoding_scope=None*)
Register an atom that will be used by the DataModel when an operation requiring data absorption is performed, like self.decode().

Parameters

- **atom** – Atom to register for absorption
- **absorb_constraints** – Constraints to be used for the absorption
- **decoding_scope** – Should be either an atom name that can be absorbed by the registered atom, or a textual description of the scope, or a list of the previous elements. If set to None, the atom will be the default one used for decoding operation if no other nodes exist with a specific scope.

show()

update_atom(*atom*)

validation_tests()
Optional test cases to validate the correct behavior of the data model

Returns `True` if the validation succeeds. `False` otherwise

Return type bool

class framework.data_model.**NodeBackend**(*data_model*)
Bases: object

__init__(*data_model*)

__module__ = 'framework.data_model'

atom_copy(*orig_atom*, *new_name=None*)

default_gen_custo = None

default_nonterm_custo = None

get_all_confs()

merge_with(*node_backend*)

prepare_atom(*atom*)

update_atom(*atom*, *existing_env=False*)

13.2.4 framework.node module

class framework.node.**BitFieldCondition**(*sf*, *val=None*, *neg_val=None*, *gt_val=None*, *lt_val=None*)
Bases: *framework.node.NodeCondition*

__init__(*sf*, *val=None*, *neg_val=None*, *gt_val=None*, *lt_val=None*)

Parameters

- **sf** (int/list of int) – subfield(s) of the BitField() on which the condition apply
- **val** (int/list of int/list of list of int) – integer(s) that satisfies the condition(s)
- **neg_val** (int/list of int/list of list of int) – integer(s) that does NOT satisfy the condition(s) (AND clause)
- **gt_val** (int/list of int/list of list of int) – condition met if subfield(s) greater than or equal to values in this field (AND clause)
- **lt_val** (int/list of int/list of list of int) – condition met if subfield(s) lesser than or equal to values in this field (AND clause)

__module__ = 'framework.node'

check(*node*)

class framework.node.**DJobGroup**(*node_list*)
Bases: object

__id__()

__init__(*node_list*)

__iter__()

__module__ = 'framework.node'

__repr__()
Return repr(self).

__reversed__()

class framework.node.**DynNode_Helpers**
Bases: object

__copy__()

__init__()

__module__ = 'framework.node'

_get_graph_info()

_update_dyn_helper(*env*)

clear_graph_info_since(*node*)

determinist = True

property graph_info

make_private(*env=None*)

reset_graph_info()

set_graph_info(*node*, *info*)

class framework.node.**Env**

Bases: object

__copy__()

__getattr__(*name*)

__init__()

__module__ = 'framework.node'

add_node_to_corrupt(*node*, *corrupt_type=None*, *corrupt_op=<function Env.<lambda>>*)

cleanup_basic_djobs(*prio*)

cleanup_remaining_djobs(*prio*)

clear_all_exhausted_nodes()

clear_exhausted_node(*node*)

property color_enabled

property delayed_jobs_pending

disable_color()

djobs_exists(*prio*)

enable_color()

execute_basic_djobs(*prio*)

exhausted_node_exists()

exhausted_nodes_amount()

get_all_djob_groups(*prio*)

get_basic_djobs(*prio*)

get_data_model()

get_djobs_by_gid(*group_id*, *prio*)

get_exhausted_nodes()

is_djob_registered(*key*, *prio*)

is_empty()

is_node_exhausted(*node*)

knowledge_source = None

notify_exhausted_node(*node*)

register_basic_djob(*func*, *args*, *prio=1*)

register_djob(*func*, *group*, *key*, *cleanup=None*, *args=None*, *prio=1*)

remove_djob(*group*, *key*, *prio*)

remove_node_to_corrupt(*node*)

set_data_model(*dm*)

update_node_refs(*node_dico*, *ignore_frozen_state*)

class framework.node.**Env4NT**

Bases: object

Define methods for non-terminal nodes

__copy__()

__init__()

__module__ = 'framework.node'

clear_drawn_node_attrs(*node_id*)

get_drawn_node_qty(*node_id*)

get_drawn_node_sz(*node_id*)

is_empty()

node_exists(*node_id*)

reset()

set_drawn_node_attrs(*node_id*, *nb*, *sz*)

update_node_ids(*id_list*)

class framework.node.**FuncCusto**(*items_to_set=None*, *items_to_clear=None*, *transform_func=None*)

Bases: *framework.node.NodeCustomization*

Function node behavior-customization To be provided to *NodeInternals.customize()*

CloneExtNodeArgs = 2

FrozenArgs = 1

__module__ = 'framework.node'

_custo_items = {1: True, 2: False}

property clone_ext_node_args_mode

property frozen_args_mode

class framework.node.**GenFuncCusto**(*items_to_set=None*, *items_to_clear=None*, *transform_func=None*)

Bases: *framework.node.NodeCustomization*

Generator node behavior-customization To be provided to *NodeInternals.customize()*

CloneExtNodeArgs = 2

ForwardConfChange = 1

ResetOnUnfreeze = 3

TriggerLast = 4

__annotations__ = {}

__module__ = 'framework.node'

_custo_items = {1: True, 2: False, 3: True, 4: False}

property clone_ext_node_args_mode

property forward_conf_change_mode

property reset_on_unfreeze_mode

property trigger_last_mode

class framework.node.IntCondition(*val=None*, *neg_val=None*, *gt_val=None*, *lt_val=None*)

Bases: *framework.node.NodeCondition*

__annotations__ = {}

__init__(*val=None*, *neg_val=None*, *gt_val=None*, *lt_val=None*)

Parameters

- **val** (int/list of int) – integer(s) that satisfies the condition
- **neg_val** (int/list of int) – integer(s) that does NOT satisfy the condition (AND clause)
- **gt_val** (*int*) – condition met if greater than or equal to this value (AND clause)
- **lt_val** (*int*) – condition met if lesser than or equal to this value (AND clause)

__module__ = 'framework.node'

check(*node*)

class framework.node.Node(*name*, *base_node=None*, *copy_dico=None*, *ignore_frozen_state=False*, *accept_external_entanglement=False*, *acceptance_set=None*, *subnodes=None*, *values=None*, *value_type=None*, *vt=None*, *new_env=False*, *description=None*)

Bases: object

A Node is the basic building-block used within a graph-based data model.

internals

Contains all the configuration of a node. A configuration is associated to the internals/contents of a node, which can live independently of the other configuration.

Type dict of str –> *NodeInternals*

current_conf

Identifier to a configuration. Every usable node use at least one main configuration, namely 'MAIN'.

Type str

name

Identifier of a node. Defined at instantiation. Shall be unique from its parent perspective.

Type str

env

One environment object is added to all the nodes of a node graph when the latter is registered within a data model (cf. DataModel.register()). It is used for sharing global resources between nodes.

Type *Env*

entangled_nodes
Collection of all the nodes entangled with this one. All the entangled nodes will react the same way as one of their peers (within some extent) if this peer is subjected to a stimuli. The node's properties related to entanglement are only the ones that directly define a node. For instance, changing a node's NodeInternals will propagate to its entangled peers but changing the state of a node's NodeInternals won't propagate. It is used for dealing with multiple instance of a same node (within the scope of a NonTerm node—cf. *NodeInternals_NonTerm.get_subnodes_with_csts()*). But this mechanism can also be used for your own specific purpose.

Type set(*Node*)

semantics
(optional) Used to associate a semantics to a node. Can be used during graph traversal in order to perform actions related to semantics.

Type *NodeSemantics*

fuzz_weight
The fuzz weight is an optional attribute of Node() which express Data Model designer's hints for prioritizing the nodes to fuzz. If set, this attribute is used by some generic *disruptors* (the ones that rely on a ModelWalker object—refer to fuzzing_primitives.py)

Type int

depth
Depth of the node within the graph from a specific given root. Will be computed lazily (only when requested).

Type int

tmp_ref_count
(internal use) Temporarily used during the creation of multiple instance of a same node, especially in order to generate unique names.

Type int

_post_freeze_handler
Is executed just after a node is frozen (which is the result of requesting its value when it is not freezed—e.g., at its creation).

Type function

CORRUPT_EXIST_COND = 5

CORRUPT_NODE_QTY = 7

CORRUPT_QTY_SYNC = 6

CORRUPT_SIZE_SYNC = 8

DEFAULT_DISABLED_NODEINT = <framework.node.NodeInternals_Empty object>

DEFAULT_DISABLED_VALUE = b''

DJOBS_PRIO_dynhelpers = 200

DJOBS_PRIO_genfunc = 300

DJOBS_PRIO_nterm_existence = 100

__copy__()

__get_confs()

__get_current_internals()

__get_internals()

__getattr__(*name*)

__getitem__(*key*)

__hash__()
Return hash(self).

__init__(*name*, *base_node=None*, *copy_dico=None*, *ignore_frozen_state=False*, *accept_external_entanglement=False*, *acceptance_set=None*, *subnodes=None*, *values=None*, *value_type=None*, *vt=None*, *new_env=False*, *description=None*)

Parameters

- **name** (*str*) – Name of the node. Every children node of a node shall have a unique name. Useful to look for specific nodes within a graph.
- **subnodes** (*list*) – (Optional) List of subnodes. If provided the Node will be created as a non-terminal node.
- **values** (*list*) – (Optional) List of strings. If provided the instantiated node will be a String-typed leaf node (taking its possible values from the parameter).
- **value_type** (VT) – (Optional) The value type that characterize the node. Defined within *value_types.py* and inherits from either *VT* or *VT_Alt*. If provided the instantiated node will be a value_type-typed leaf node.
- **vt** (VT) – alias to *value_type*.
- **base_node** (Node) – (Optional) If provided, it will be used as a template to create the new node.
- **ignore_frozen_state** (*bool*) – [If *base_node* provided] If True, the clone process of base_node will ignore its current state.
- **accept_external_entanglement** (*bool*) – [If *base_node* provided] If True, during the cloning process of base_node, every entangled nodes outside the current graph will be referenced within the new node without being copied. Otherwise, a *Warning* message will be raised.
- **acceptance_set** (*set*) – [If *base_node* provided] If provided, will be used as a set of entangled nodes that could be referenced within the new node during the cloning process.
- **copy_dico** (*dict*) – [If *base_node* provided] It is used internally during the cloning process, and should not be used for any functional purpose.
- **new_env** (*bool*) – [If *base_node* provided] If True, the *base_node* attached *Env()* will be copied. Otherwise, the same will be used. If *ignore_frozen_state* is True, a new *Env()* will be used.
- **description** (*str*) – textual description of the node

__lt__(*other*)
Return self<value.

__module__ = 'framework.node'

__set_current_internals__(*internal*)

__setitem__(*key*, *val*)

__str__()
Return str(self).

_check_conf(*conf*)

_compute_confs(*conf*, *recursive*)

_finalize_nonterm_node(*conf*, *depth=None*)

_get_all_paths_rec(*pname*, *htable*, *conf*, *recursive*, *first=True*, *resolve_generator=False*, *clone_idx=0*)

_get_value(*conf=None*, *recursive=True*, *return_node_internals=False*, *restrict_csp=False*)

_post_freeze(*node_internals*, *wrapping_node*, *next_conf*, *recursive*, *return_node_internals*)

static _print(*msg*, *rgb*, *style=''*, *nl=True*, *log_func=<built-in method write of _io.TextIOWrapper object>*, *pretty_print=True*)

static _print_contents(*msg*, *style=''*, *nl=True*, *log_func=<built-in method write of _io.TextIOWrapper object>*, *pretty_print=True*)

static _print_name(*msg*, *style=''*, *nl=True*, *log_func=<built-in method write of _io.TextIOWrapper object>*, *pretty_print=True*)

static _print_nonterm(*msg*, *style=\x1b[1m'*, *nl=True*, *log_func=<built-in method write of _io.TextIOWrapper object>*, *pretty_print=True*)

static _print_raw(*msg*, *style=''*, *nl=True*, *hlight=False*, *log_func=<built-in method write of _io.TextIOWrapper object>*, *pretty_print=True*)

static _print_type(*msg*, *style=\x1b[1m'*, *nl=True*, *log_func=<built-in method write of _io.TextIOWrapper object>*, *pretty_print=True*)

_reset_depth(*parent_depth*)

_set_clone_info(*info*, *node*)
Used to propagate random draw results when a NonTerm node is frozen to the dynamic nodes of its attached subgraphs, namely GenFunc/Func nodes which are the only ones which can act dynamically.

_set_subtrees_current_conf(*node*, *conf*, *reverse*, *ignore_entanglement=False*)

_tobytes(*conf=None*, *recursive=True*)

absorb(*blob*, *constraints=AbsCsts()*, *conf=None*, *pending_postpone_desc=None*)

add_conf(*conf*)

property c
Property linked to *self.internals* (read only)

property cc
Property linked to the current node's *internals* (read / write)

clear_attr(*name*, *conf=None*, *all_conf=False*, *recursive=False*)

property color_enabled

compliant_with(*internals_criteria=None*, *semantics_criteria=None*, *conf=None*)

conf(*conf=None*)

property confs
Property giving all node's configurations (read only)

property debug

disable_color()

enable_color()

enforce_absorb_constraints(*csts*, *conf=None*)

entangle_with(*node*)

static filter_out_entangled_nodes(*node_list*)

fix_synchronized_nodes(*conf=None*)

freeze(*conf=None*, *recursive=True*, *return_node_internals=False*, *restrict_csp=False*, *resolve_csp=False*)

Parameters

- **conf** –
- **recursive** –
- **return_node_internals** –
- **restrict_csp** – Only effective when a CSP is part of the data description. When set to True, if the node on which this method is called is a variable of the CSP, then its domain will be shrunk to its current value. Thus, the node won't change when the CSP will be resolved.
- **resolve_csp** – Only effective when a CSP is part of the data description. When set to True, the CSP will be resolved and the data generated will comply with the solution.

Returns:

gather_alt_confs()

get_all_paths(*conf=None*, *recursive=True*, *depth_min=None*, *depth_max=None*, *resolve_generator=False*, *flush_cache=True*)

Parameters **resolve_generator** – if *True*, the generator nodes will be resolved in order to perform the search within. But there could be side-effects on the graph, because some parts of the graph could end up frozen if they are used as generator parameters. If *False*, generator nodes won't be resolved, but they could already be in a resolved state before this method is called on them. It means that no side effects could result from the call of this method. And thus for this latter case, the method works as if *resolve_generator* is set to *True*.

Returns

the keys are either a 'path' or a tuple ('path', int) when the path already exists (case of the same node used more than once within the same non-terminal)

Return type dict

get_all_paths_from(*node*, *conf=None*, *flush_cache=True*, *resolve_generator=False*)

get_clone(*name=None*, *ignore_frozen_state=False*, *accept_external_entanglement=False*, *acceptance_set=None*, *new_env=True*)

Create a new node. To be used within a graph-based data model.

Parameters

- **name** (*str*) – name of the new Node instance. If None the current name will be used.
- **ignore_frozen_state** (*bool*) – if set to False, the clone function will produce a Node with the same state as the duplicated Node. Otherwise, only the state won't be kept.
- **accept_external_entanglement** (*bool*) – refer to the corresponding Node parameter

- **acceptance_set** (*set*) – refer to the corresponding Node parameter
- **new_env** (*bool*) – If True, the current *Env()* will be copied. Otherwise, the same will be used.

Returns duplicated Node object

Return type *Node*

get_csp()

get_current_conf()

get_env()

get_first_node_by_path(*path_regexp*, *conf=None*, *flush_cache=True*, *resolve_generator=False*)
Return the first Node that match the *path_regexp* parameter.

Parameters

- **path_regexp** (*str*) – path regexp of the requested nodes
- **conf** (*str*) – Node configuration to use for the search
- **flush_cache** (*bool*) – If False, and a previous search has been performed, the outcomes will be used for this one, which will improve the performance.

Returns the first Node that match the path regexp

Return type *Node*

get_fuzz_weight()
Return the fuzzing weight of the node.

Returns the fuzzing weight

Return type int

get_internals_backup()

get_nodes_by_paths(*path_list*)
Provide a dictionnary of the nodes referenced by the paths provided in @path_list. Keys of the dict are the paths provided in @path_list.

Parameters **path_list** – list of paths referencing nodes of interest

Returns dictionary mapping path to nodes

Return type dict

get_nodes_names(*conf=None*, *verbose=False*, *terminal_only=False*, *flush_cache=True*)

get_path_from(*node*, *conf=None*, *flush_cache=True*, *resolve_generator=False*)

get_private(*conf=None*)

get_reachable_nodes(*internals_criteria=None*, *semantics_criteria=None*, *owned_conf=None*, *conf=None*, *path_regexp=None*, *exclude_self=False*, *respect_order=False*, *top_node=None*, *ignore_fstate=False*, *resolve_generator=False*, *relative_depth=- 1*)

Parameters

- **internals_criteria** –
- **semantics_criteria** –
- **owned_conf** –

- **conf** –
- **path_regexp** –
- **exclude_self** –
- **respect_order** –
- **top_node** –
- **ignore_fstate** –
- **resolve_generator** – if *True*, the generator nodes will be resolved in order to perform the search within. But there will be side-effects on the graph, because some parts of the graph could end up frozen if they are used as generator parameters. If *False*, generator nodes won't be resolved, but they could already be in a resolved state before this method is called on them. It means that no side effects could result from the call of this method. And thus for this latter case, the method works as if *resolve_generator* is set to *True*.
- **relative_depth** – For internal use only

Returns:

get_semantics()

get_value(*conf=None*, *recursive=True*, *return_node_internals=False*, *restrict_csp=False*, *resolve_csp=False*)

Parameters

- **conf** –
- **recursive** –
- **return_node_internals** –
- **restrict_csp** – Only effective when a CSP is part of the data description. When set to True, if the node on which this method is called is a variable of the CSP, then its domain will be shrunk to its current value. Thus, the node won't change when the CSP will be resolved.
- **resolve_csp** – Only effective when a CSP is part of the data description. When set to True, the CSP will be resolved and the data generated will comply with the solution.

Returns:

is_attr_set(*name*, *conf=None*)

is_conf_existing(*conf*)

is_empty(*conf=None*)

is_exhausted(*conf=None*)

is_frozen(*conf=None*)

is_func(*conf=None*)

is_genfunc(*conf=None*)

is_nonterm(*conf=None*)

is_path_valid(*path*, *resolve_generator=False*)

is_term(*conf=None*)

is_typed_value(*conf=None*, *subkind=None*)

iter_nodes_by_path(*path_regexp*, *conf=None*, *flush_cache=True*, *resolve_generator=False*)
iterate over all the nodes that match the *path_regexp* parameter.

Note: the set of nodes that is used to perform the search include the node itself and all the subnodes behind it.

Parameters

- **path_regexp** (*str*) – path regexp of the requested nodes
- **conf** (*str*) – Node configuration to use for the search
- **flush_cache** (*bool*) – If False, and a previous search has been performed, the outcomes will be used for this one, which will improve the performance.

Returns generator of the nodes that match the path regexp

iter_paths(*conf=None*, *recursive=True*, *depth_min=None*, *depth_max=None*, *only_paths=False*, *resolve_generator=False*, *flush_cache=True*)

make_determinist(*conf=None*, *all_conf=False*, *recursive=False*)

make_empty(*conf=None*)

make_finite(*conf=None*, *all_conf=False*, *recursive=False*)

make_infinite(*conf=None*, *all_conf=False*, *recursive=False*)

make_random(*conf=None*, *all_conf=False*, *recursive=False*)

make_synchronized_with(*scope*, *node=None*, *param=None*, *sync_obj=None*, *conf=None*)

property no_more_solution_for_csp

pretty_print(*max_size=None*, *conf=None*)

register_post_freeze_handler(*func*)

remove_conf(*conf*)

reset_fuzz_weight(*recursive=False*)
Reset to standard (1) the fuzzing weight that is associated to this node, and all its subnodes if *recursive* parameter is set to *True*.

Parameters **recursive** (*bool*) – if set to *True*, reset also every subnodes (all reachable nodes from this one).

Returns None

reset_state(*recursive=False*, *exclude_self=False*, *conf=None*, *ignore_entanglement=False*)

set_absorb_helper(*helper*, *conf=None*)

set_attr(*name*, *conf=None*, *all_conf=False*, *recursive=False*)

set_contents(*base_node*, *copy_dico=None*, *ignore_frozen_state=False*, *accept_external_entanglement=False*, *acceptance_set=None*, *preserve_node=True*)
Set the contents of the node based on the one provided within *base_node*. This method performs a deep copy of *base_node*, but some parameters can change the behavior of the copy.

Note: python deepcopy() is not used for performance reason (10 to 20 times slower) and as it does not work for all cases.

Parameters

- **base_node** (Node) – (Optional) Used as a template to create the new node.
- **ignore_frozen_state** (*bool*) – If True, the clone process of base_node will ignore its current state.
- **preserve_node** (*bool*) – preserve the *NodeInternals* attributes (making sense to preserve) of the possible overwritten NodeInternals.
- **accept_external_entanglement** (*bool*) – If True, during the cloning process of base_node, every entangled nodes outside the current graph will be referenced within the new node without being copied. Otherwise, a *Warning* message will be raised.
- **acceptance_set** (*set*) – If provided, will be used as a set of entangled nodes that could be referenced within the new node during the cloning process.
- **copy_dico** (*dict*) – It is used internally during the cloning process, and should not be used for any functional purpose.

Returns For each subnodes of *base_node* (keys), reference the corresponding subnodes within the new node.

Return type dict

set_csp(*csp:* framework.constraint_helpers.CSP)

set_current_conf(*conf*, *recursive=True*, *reverse=False*, *root_regexp=None*, *ignore_entanglement=False*)

set_default_value(*value*, *conf=None*)

set_env(*env*)

set_frozen_value(*value*, *conf=None*)

set_func(*func*, *func_node_arg=None*, *func_arg=None*, *conf=None*, *ignore_entanglement=False*, *provide_helpers=False*, *preserve_node=True*)

set_fuzz_weight(*w*)

Set the fuzzing weight of the node to *w*.

The fuzz weight is an optional attribute of Node() which express Data Model designer's hints for prioritizing the nodes to fuzz. If set, this attribute is used by some generic *disruptors* (the ones that rely on a ModelWalker object—refer to fuzzing_primitives.py)

Parameters **w** (*int*) – Value of the weight (by default every nodes has a weight of 1)

Returns None

set_generator_func(*gen_func*, *func_node_arg=None*, *func_arg=None*, *conf=None*, *ignore_entanglement=False*, *provide_helpers=False*, *preserve_node=True*)

set_internals(*backup*)

set_private(*val*, *conf=None*)

set_semantics(*sem*)

set_size_from_constraints(*size=None*, *encoded_size=None*, *conf=None*)

set_subnodes_basic(*node_list*, *conf=None*, *ignore_entanglement=False*, *separator=None*, *preserve_node=True*)

set_subnodes_full_format(*subnodes_order*, *subnodes_attrs*, *conf=None*, *separator=None*, *preserve_node=True*)

set_subnodes_with_csts(*wlnode_list*, *conf=None*, *ignore_entanglement=False*, *separator=None*, *preserve_node=True*)

set_values(*values=None*, *value_type=None*, *conf=None*, *ignore_entanglement=False*, *preserve_node=True*)

show(*conf=None*, *verbose=True*, *print_name_func=None*, *print_contents_func=None*, *print_raw_func=None*, *print_nonterm_func=None*, *print_type_func=None*, *alpha_order=False*, *raw_limit=None*, *log_func=<built-in method write of _io.TextIOWrapper object>*, *pretty_print=True*, *display_title=True*, *display_gen_node=True*)

synchronized_with(*scope*, *conf=None*)

to_ascii(*conf=None*, *recursive=True*)

to_bytes(*conf=None*, *recursive=True*)

to_formatted_str(*conf=None*, *recursive=True*)

to_str(*conf=None*, *recursive=True*)

unfreeze(*conf=None*, *recursive=True*, *dont_change_state=False*, *ignore_entanglement=False*, *only_generators=False*, *reevaluate_constraints=False*, *walk_csp=False*, *walk_csp_step_size=1*)

unfreeze_all(*recursive=True*, *ignore_entanglement=False*)

update(*node_update_dict*, *stop_on_error=True*)

walk(*conf=None*, *recursive=True*, *steps_num=1*)

class framework.node.**NodeAbstraction**

Bases: object

This class can be used in place of an node_arg for Func and GenFunc Nodes. It enables you to define in your data model higher level classes upon Nodes to facilitate Nodes manipulation within Func and GenFunc Nodes, with regards to your data model paradigm.

__module__ = 'framework.node'

get_concrete_nodes()

Shall return an Node or a list of Nodes

make_private()

This method is called during Node copy process. It aims to make all your metadata private (if needed). Note that you don't have to deal with your Nodes.

set_concrete_nodes(*nodes_args*)

Shall save an Node or a list of Nodes (depending on what returns get_concrete_nodes())

class framework.node.**NodeCondition**

Bases: object

Base class for every node-related conditions. (Note that NodeCondition may be copied many times. If some attributes need to be fully copied, handle this through __copy__() overriding).

__annotations__ = {}

__module__ = 'framework.node'

_check_inclusion(*curr_val*, *val=None*, *neg_val=None*)

_check_int(*val*, *gt_val=None*, *lt_val=None*)

check(*node*)

class framework.node.**NodeCustomization**(*items_to_set=None*, *items_to_clear=None*, *transform_func=None*)

Bases: object

Base class for node cutomization

__annotations__ = {}

__copy__()

__getitem__(*key*)

__init__(*items_to_set=None*, *items_to_clear=None*, *transform_func=None*)

__module__ = 'framework.node'

_custo_items = {}

clear_items(*items_to_clear*)

copy_from(*node_custo*)

set_items(*items_to_set*)

property transform_func

class framework.node.NodeInternals(*arg=None*)

Bases: object

Base class for implementing the contents of a node.

Abs_Postpone = 6

AutoSeparator = 16

DEBUG = 40

DISABLED = 100

Determinist = 3

Finite = 4

Freezable = 1

Highlight = 30

LOCKED = 50

Mutable = 2

Separator = 15

__hash__()

Return hash(self).

__init__(*arg=None*)

__module__ = 'framework.node'

_clear_attr_direct(*name*)

_get_value(*conf=None*, *recursive=True*, *return_node_internals=False*, *restrict_csp=False*)

_init_specific(*arg*)

_make_private_specific(*ignore_frozen_state*, *accept_external_entanglement*)

_make_specific(*name*)

_match_mandatory_attrs(*criteria*)

_match_mandatory_custo(*criteria*)

_match_negative_attrs(*criteria*)

_match_negative_custo(*criteria*)

_match_negative_node_kinds(*criteria*)

_match_negative_node_subkinds(*criteria*)

_match_node_constraints(*criteria*)

_match_node_kinds(*criteria*)

_match_node_subkinds(*criteria*)

_set_attr_direct(*name*)

_unmake_specific(*name*)

_update_node_refs(*node_dico*, *debug*)

absorb(*blob*, *constraints*, *conf*, *pending_postpone_desc=None*)

clear_attr(*name*)

clear_child_attr(*name*, *conf=None*, *all_conf=False*, *recursive=False*)

clear_clone_info_since(*node*)
Cleanup obsolete graph internals information prior to what has been registered with the node given as parameter.

customize(*custo*)

property debug

default_custo = None

enforce_absorb_constraints(*csts*)

property env

get_attrs_copy()

get_current_subkind()

get_node_sync(*scope*)

get_private()

get_raw_value(***kwargs*)

has_subkinds()

property highlight

is_attr_set(*name*)

is_exhausted()

is_frozen()

make_private(*ignore_frozen_state*, *accept_external_entanglement*, *delayed_node_internals*, *forget_original_sync_objs=False*)

match(*internals_criteria*)

pretty_print(*max_size=None*)

reset_depth_specific(*depth*)

set_absorb_helper(*helper*)

set_attr(*name*)

set_attrs_from(*all_attrs*)

set_child_attr(*name*, *conf=None*, *all_conf=False*, *recursive=False*)

set_clone_info(*info*, *node*)
Report to Node._set_clone_info() some information about graph internals

set_contents_from(*node_internals*)

set_node_sync(*scope*, *node=None*, *param=None*, *sync_obj=None*)

set_private(*val*)

set_size_from_constraints(*size*, *encoded_size*)

synchronize_nodes(*src_node*)

class framework.node.**NodeInternalsCriteria**(*mandatory_attrs=None*, *negative_attrs=None*, *node_kinds=None*, *negative_node_kinds=None*, *node_subkinds=None*, *negative_node_subkinds=None*, *mandatory_custo=None*, *negative_custo=None*, *required_csts=None*, *negative_csts=None*)

Bases: object

__init__(*mandatory_attrs=None*, *negative_attrs=None*, *node_kinds=None*, *negative_node_kinds=None*, *node_subkinds=None*, *negative_node_subkinds=None*, *mandatory_custo=None*, *negative_custo=None*, *required_csts=None*, *negative_csts=None*)

__module__ = 'framework.node'

_handle_user_input(*crit*)

clear_node_constraint(*cst*)

extend(*ic*)

get_all_node_constraints()

get_node_constraint(*cst*)

has_node_constraints()

set_node_constraint(*cst*, *required*)

class framework.node.**NodeInternals_Empty**(*arg=None*)

Bases: *framework.node.NodeInternals*

__annotations__ = {}

__module__ = 'framework.node'

_get_value(*conf=None*, *recursive=True*, *return_node_internals=False*, *restrict_csp=False*)

get_child_nodes_by_attr(*internals_criteria*, *semantics_criteria*, *owned_conf*, *conf*, *path_regexp*, *exclude_self*, *respect_order*, *relative_depth*, *top_node*, *ignore_fstate*, *resolve_generator=False*)

get_raw_value(***kwargs*)

set_child_env(*env*)

class framework.node.**NodeInternals_Func**(*arg=None*)

Bases: *framework.node.NodeInternals_Term*

__annotations__ = {}

__get_value_specific_mode1(*conf*, *recursive*)
In mode1, we freeze 'node_arg' attribute and give the value to the function

__get_value_specific_mode2(*conf*, *recursive*)
In mode2, we give the 'node_arg' to the function and let it do whatever it wants

__module__ = 'framework.node'

_get_value_specific(*conf*, *recursive*)

_init_specific(*arg*)

_make_private_term_specific(*ignore_frozen_state*, *accept_external_entanglement*)

_reset_state_specific(*recursive*, *exclude_self*, *conf*, *ignore_entanglement*)

_unfreeze_reevaluate_constraints(*current_val*)

_unfreeze_without_state_change(*current_val*)

absorb(*blob*, *constraints*, *conf*, *pending_postpone_desc=None*)

cancel_absorb()

clear_clone_info_since(*node*)
Cleanup obsolete graph internals information prior to what has been registered with the node given as parameter.

confirm_absorb()

customize(*custo*)

default_custo = <framework.node.FuncCusto object>

get_node_args()

import_func(*fct*, *fct_node_arg=None*, *fct_arg=None*, *provide_helpers=False*)

make_args_private(*node_dico*, *entangled_set*, *ignore_frozen_state*, *accept_external_entanglement*)

set_clone_info(*info*, *node*)
Report to Node._set_clone_info() some information about graph internals

set_func_arg(*node=None*, *fct_arg=None*)

set_size_from_constraints(*size*, *encoded_size*)

class framework.node.**NodeInternals_GenFunc**(*arg=None*)
Bases: *framework.node.NodeInternals*

__annotations__ = {}

__getattr__(*name*)

__module__ = 'framework.node'

_get_delayed_value(*conf=None*, *recursive=True*, *restrict_csp=False*)

_get_value(*conf=None*, *recursive=True*, *return_node_internals=False*, *restrict_csp=False*)

_init_specific(*arg*)

_make_private_specific(*ignore_frozen_state*, *accept_external_entanglement*)

_make_specific(*name*)

_unmake_specific(*name*)

absorb(*blob*, *constraints*, *conf*, *pending_postpone_desc=None*)

cancel_absorb()

clear_child_attr(*name*, *conf=None*, *all_conf=False*, *recursive=False*)

clear_clone_info_since(*node*)
Cleanup obsolete graph internals information prior to what has been registered with the node given as parameter.

confirm_absorb()

default_custo = <framework.node.GenFuncCusto object>

property env

property generated_node

get_child_all_path(*name*, *htable*, *conf*, *recursive*, *resolve_generator=False*)

get_child_nodes_by_attr(*internals_criteria*, *semantics_criteria*, *owned_conf*, *conf*, *path_regexp*, *exclude_self*, *respect_order*, *relative_depth*, *top_node*, *ignore_fstate*, *resolve_generator=False*)

get_node_args()

get_raw_value(***kwargs*)

import_generator_func(*generator_func*, *generator_node_arg=None*, *generator_arg=None*, *provide_helpers=False*)

is_exhausted()

is_frozen()

make_args_private(*node_dico*, *entangled_set*, *ignore_frozen_state*, *accept_external_entanglement*)

reset_depth_specific(*depth*)

reset_fuzz_weight(*recursive*)

reset_generator()

reset_state(*recursive=False*, *exclude_self=False*, *conf=None*, *ignore_entanglement=False*)

set_child_attr(*name*, *conf=None*, *all_conf=False*, *recursive=False*)

set_child_current_conf(*node*, *conf*, *reverse*, *ignore_entanglement*)

set_child_env(*env*)

set_clone_info(*info*, *node*)
Report to Node._set_clone_info() some information about graph internals

set_generator_func_arg(*generator_node_arg=None*, *generator_arg=None*)

set_size_from_constraints(*size*, *encoded_size*)

unfreeze(*conf=None*, *recursive=True*, *dont_change_state=False*, *ignore_entanglement=False*, *only_generators=False*, *reevaluate_constraints=False*)

unfreeze_all(*recursive=True*, *ignore_entanglement=False*)

class framework.node.**NodeInternals_NonTerm**(*arg=None*)
Bases: *framework.node.NodeInternals*

It is a kind of node internals that enable to structure the graph through a specific grammar...

INFINITY_LIMIT = 30

class NodeAttrs
Bases: object

__copy__()

__module__ = 'framework.node'

_current_qty = None

_default_qty = None

_max = None

_min = None

_planned_reset = False

_previous_current_qty_was_none = False

_previous_qty = None

_qty_sequence = None

property current_qty

property default_qty

exhausted_seq = False

next_qty()

perform_planned_reset()

plan_reset()

property qty

property qty_sequence

reset()

unplan_reset()

unroll()

__annotations__ = {}

__iter_csts(*node_list*)

__iter_csts_verbose(*node_list*)

__module__ = 'framework.node'

static _cleanup_delayed_nodes(*node*, *node_list*, *idx*, *conf*, *rec*)

_cleanup_entangled_nodes()

_cleanup_entangled_nodes_from(*node*)

_clear_drawn_node_attrs(*node*)

_clone_node(*base_node*, *node_no*, *force_clone=False*, *ignore_frozen_state=True*)

_clone_node_cleanup()

_clone_separator(*sep_node*, *unique*, *force_clone=False*, *ignore_frozen_state=True*)

_clone_separator_cleanup()

_construct_subnodes(*node_desc*, *subnode_list*, *mode*, *ignore_sep_fstate*, *ignore_separator=False*, *lazy_mode=True*)

_copy_nodelist(*node_list*)

static _existence_from_node(*node*)

static _expand_delayed_nodes(*node*, *node_list*, *idx*, *conf*, *rec*)

_get_heavier_component(*comp_list*, *check_existence=False*)

_get_info_from_subnode_description(*node_desc*)

static _get_next_heavier_component(*comp_list*, *excluded_idx*)

static _get_next_random_component(*comp_list*, *excluded_idx*, *seed=None*)

_get_node_and_minmax_from(*node_desc*)

_get_node_from(*node_desc*)

_get_random_component(*comp_list*, *total_weight*, *check_existence=False*)

_get_value(*conf=None*, *recursive=True*, *after_encoding=True*, *return_node_internals=False*, *restrict_csp=False*)

The parameter return_node_internals is not used for non terminal nodes, only for terminal nodes. However, keeping it also for non terminal nodes avoid additional checks in the code.

_init_specific(*arg*)

_make_private_specific(*ignore_frozen_state*, *accept_external_entanglement*)

_make_specific(*name*)

_parse_node_desc(*node_desc*)

_precondition_subnode_ops()

static _qty_from_node(*node*)

_reset_state_info(*new_info=None*, *nodes_drawn_qty=None*)

_set_drawn_node_attrs(*node*, *nb*, *sz*)

static _size_from_node(*node*, *for_encoded_size=False*)

_unmake_specific(*name*)

absorb(*blob*, *constraints*, *conf*, *pending_postpone_desc=None*)

TOFIX: Checking existence condition independently from data description order is not supported. Only supported within the same non-terminal node. Use delayed job infrastructure to cover all cases (TBC).

add(*node*, *min=1*, *max=1*, *default_qty=None*, *after=None*, *before=None*, *idx=None*)

This method add a new node to this non-terminal. The location and the quantity can be configured through the parameters.

Parameters

- **node** (Node) – The node to add
- **min** – The minimum number of repetition of this node within the non-terminal node
- **max** – The maximum number of repetition of this node within the non-terminal node
- **default_qty** – the default number of repetition of this node within the non-terminal node
- **after** – If not None, it should be the node (within the non-terminal) *after* which the new node will be inserted.

- **before** – If not None, it should be the node (within the non-terminal) *before* which the new node will be inserted.
- **idx** – If not None, it should provide the position in the list of subnodes where the new node will be inserted.

cancel_absorb()

change_subnodes_csts(*csts_ch*)

clear_child_attr(*name*, *conf=None*, *all_conf=False*, *recursive=False*)

clear_clone_info_since(*node*)
Cleanup obsolete graph internals information prior to what has been registered with the node given as parameter.

confirm_absorb()

default_custo = <framework.node.NonTermCusto object>

static existence_corrupt_hook(*node*, *exist*)

flatten_node_list(*node_list*)
Return a list of the form: [subnode1, subnode2, subnode3, ...] In case of Pick-type sections within the parent node, sublists are included within the previous one and include the alternative subnodes, so that the list looks like: [subnode1, [snode21, snode22, ...], subnode3, ...]

Parameters node_list –

Returns:

get_child_all_path(*name*, *htable*, *conf*, *recursive*, *resolve_generator=False*)

get_child_nodes_by_attr(*internals_criteria*, *semantics_criteria*, *owned_conf*, *conf*, *path_regexp*, *exclude_self*, *respect_order*, *relative_depth*, *top_node*, *ignore_fstate*, *resolve_generator=False*)

get_drawn_node_qty(*node_ref*)

get_raw_value(***kwargs*)

get_separator_node()

get_subnode(*num*)

get_subnode_default_qty(*node*)

get_subnode_idx(*node*)

get_subnode_minmax(*node*)

get_subnode_off(*num*)

get_subnode_qty()

get_subnodes_collection()

get_subnodes_csts_copy(*node_dico=None*)

get_subnodes_with_csts()
Generate the structure of the non terminal node.

import_subnodes_basic(*node_list*, *separator=None*, *preserve_node=False*)

import_subnodes_full_format(*subnodes_order=None*, *subnodes_attrs=None*, *frozen_node_list=None*, *current_flat_nodelist=None*, *internals=None*, *nodes_drawn_qty=None*, *custo=None*, *exhaust_info=None*, *separator=None*)

import_subnodes_with_csts(*wlnode_list*, *separator=None*, *preserve_node=False*)

is_exhausted()

is_frozen()

make_private_subnodes(*node_dico*, *func_nodes*, *env*, *ignore_frozen_state*, *accept_external_entanglement*, *entangled_set*, *delayed_node_internals*)

static nodeqty_corrupt_hook(*node*, *mini*, *maxi*)

static qtysync_corrupt_hook(*node*, *qty*)

replace_subnode(*old*, *new*)

reset(*nodes_drawn_qty=None*, *custo=None*, *exhaust_info=None*, *preserve_node=False*)

reset_depth_specific(*depth*)

reset_fuzz_weight(*recursive*)

reset_state(*recursive=False*, *exclude_self=False*, *conf=None*, *ignore_entanglement=False*)

set_child_attr(*name*, *conf=None*, *all_conf=False*, *recursive=False*)

set_child_current_conf(*node*, *conf*, *reverse*, *ignore_entanglement*)

set_child_env(*env*)

set_clone_info(*info*, *node*)
Report to Node._set_clone_info() some information about graph internals

set_encoder(*encoder*)

set_separator_node(*sep_node*, *prefix=True*, *suffix=True*, *unique=False*, *always=False*)

set_size_from_constraints(*size*, *encoded_size*)

set_subnode_default_qty(*node*, *default_qty=None*)

set_subnode_minmax(*node*, *min=None*, *max=None*)

static sizesync_corrupt_hook(*node*, *length*)

structure_will_change()
To be used only in Finite mode. Return True if the structure will change the next time _get_value() will be called.

Returns: bool

unfreeze(*conf=None*, *recursive=True*, *dont_change_state=False*, *ignore_entanglement=False*, *only_generators=False*, *reevaluate_constraints=False*)

unfreeze_all(*recursive=True*, *ignore_entanglement=False*)

class framework.node.**NodeInternals_Term**(*arg=None*)
Bases: *framework.node.NodeInternals*

__annotations__ = {}

__module__ = 'framework.node'

static _convert_to_internal_repr(*val*)

_get_value(*conf=None*, *recursive=True*, *return_node_internals=False*, *restrict_csp=False*)

_get_value_specific(*conf*, *recursive*)

_init_specific(*arg*)

_make_private_specific(*ignore_frozen_state*, *accept_external_entanglement*)

_make_private_term_specific(*ignore_frozen_state*, *accept_external_entanglement*)

_reset_state_specific(*recursive*, *exclude_self*, *conf*, *ignore_entanglement*)

_set_default_value(*val*)

_set_default_value_specific(*val*)

_set_frozen_value(*val*)

_unfreeze_reevaluate_constraints(*current_val*)

_unfreeze_without_state_change(*current_val*)

_update_value_specific(*value*)

absorb(*blob*, *constraints*, *conf*, *pending_postpone_desc=None*)

absorb_auto_helper(*blob*, *constraints*)

cancel_absorb()

confirm_absorb()

do_absorb(*blob*, *constraints*, *off*, *size*)

do_cleanup_absorb()

do_revert_absorb()

get_child_all_path(*name*, *htable*, *conf*, *recursive*, *resolve_generator=False*)

get_child_nodes_by_attr(*internals_criteria*, *semantics_criteria*, *owned_conf*, *conf*, *path_regexp*, *exclude_self*, *respect_order*, *relative_depth*, *top_node*, *ignore_fstate*, *resolve_generator=False*)

get_raw_value(***kwargs*)

is_exhausted()

is_frozen()

reset_depth_specific(*depth*)

reset_fuzz_weight(*recursive*)

reset_state(*recursive=False*, *exclude_self=False*, *conf=None*, *ignore_entanglement=False*)

set_child_current_conf(*node*, *conf*, *reverse*, *ignore_entanglement*)

set_child_env(*env*)

unfreeze(*conf=None*, *recursive=True*, *dont_change_state=False*, *ignore_entanglement=False*, *only_generators=False*, *reevaluate_constraints=False*)

unfreeze_all(*recursive=True*, *ignore_entanglement=False*)

update_value(*value*)

class framework.node.**NodeInternals_TypedValue**(*arg=None*)

Bases: *framework.node.NodeInternals_Term*

__annotations__ = {}

__getattr__(*name*)

__module__ = 'framework.node'

_get_value_specific(*conf=None*, *recursive=True*)

_init_specific(*arg*)

_make_private_term_specific(*ignore_frozen_state*, *accept_external_entanglement*)

_make_specific(*name*)

_reset_state_specific(*recursive*, *exclude_self*, *conf*, *ignore_entanglement*)

_set_default_value_specific(*val*)

_unfreeze_reevaluate_constraints(*current_val*)

_unfreeze_without_state_change(*current_val*)

_unmake_specific(*name*)

_update_value_specific(*value*)

absorb_auto_helper(*blob*, *constraints*)

do_absorb(*blob*, *constraints*, *off*, *size*)

do_cleanup_absorb()

do_revert_absorb()

get_current_subkind()

get_raw_value(***kwargs*)

get_specific_fuzzy_values()

get_value_type()

has_subkinds()

import_value_type(*value_type*)

is_exhausted()

pretty_print(*max_size=None*)

set_size_from_constraints(*size*, *encoded_size*)

set_specific_fuzzy_values(*vals*)

class framework.node.**NodeSemantics**(*attrs=None*)

Bases: object

To be used while defining a data model as a means to associate semantics to an Node.

__init__(*attrs=None*)

__module__ = 'framework.node'

__str__()

Return str(self).

_match_exclusive_criteria(*criteria*)

_match_mandatory_criteria(*criteria*)

_match_negative_criteria(*criteria*)

_match_optionalbut1_criteria(*criteria*)

add_attributes(*attrs*)

make_private()
This method is called during Node copy process. It aims to make all your metadata private (if needed).

match(*semantics_criteria*)
This method is called within get_reachable_nodes() (when the 'semantics' parameter is provided) to select Node that match the given semantics.

what_match_from(*raw_criteria_list*)

class framework.node.**NodeSemanticsCriteria**(*optionalbut1_criteria=None*, *mandatory_criteria=None*, *exclusive_criteria=None*, *negative_criteria=None*)
Bases: object

__bool__()

__init__(*optionalbut1_criteria=None*, *mandatory_criteria=None*, *exclusive_criteria=None*, *negative_criteria=None*)

__module__ = 'framework.node'

_handle_user_input(*crit*)

extend(*sc*)

get_exclusive_criteria()

get_mandatory_criteria()

get_negative_criteria()

get_optionalbut1_criteria()

set_exclusive_criteria(*criteria*)

set_mandatory_criteria(*criteria*)

set_negative_criteria(*criteria*)

set_optionalbut1_criteria(*criteria*)

class framework.node.**NodeSeparator**(*node*, *prefix=True*, *suffix=True*, *unique=False*, *always=False*)
Bases: object

A node separator is used (optionnaly) by a non-terminal node as a separator between each subnode.

make_private
used for full copy

Type function

__init__(*node*, *prefix=True*, *suffix=True*, *unique=False*, *always=False*)

Parameters

- **node** (Node) – node to be used for separation.
- **prefix** (*bool*) – if *True*, a serapator will also be placed at the begining.
- **suffix** (*bool*) – if *True*, a serapator will also be placed at the end.
- **unique** (*bool*) – if *False*, the same node will be used for each separation, otherwise a new node will be generated.
- **always** (*bool*) – if *True*, the separator will be always generated even if the subnodes it separates are not generated because their evaluated quantity is 0.

__module__ = 'framework.node'

make_private(*node_dico*, *ignore_frozen_state*)

class framework.node.**NonTermCusto**(*items_to_set=None*, *items_to_clear=None*, *transform_func=None*)
Bases: *framework.node.NodeCustomization*

Non-terminal node behavior-customization To be provided to *NodeInternals.customize()*

CollapsePadding = 4

CycleClone = 2

DelayCollapsing = 5

FrozenCopy = 3

FullCombinatory = 6

MutableClone = 1

StickToDefault = 7

__annotations__ = {}

__module__ = 'framework.node'

_custo_items = {1: True, 2: False, 3: True, 4: False, 5: False, 6: False, 7: False}

property collapse_padding_mode

property cycle_clone_mode

property delay_collapsing

property frozen_copy_mode

property full_combinatory_mode

property mutable_clone_mode

property stick_to_default_mode

class framework.node.**RawCondition**(*val=None*, *neg_val=None*, *cond_func=None*, *case_sensitive=True*)
Bases: *framework.node.NodeCondition*

__annotations__ = {}

__init__(*val=None*, *neg_val=None*, *cond_func=None*, *case_sensitive=True*)

Parameters

- **val** (bytes/list of bytes) – value(s) that satisfies the condition
- **neg_val** (bytes/list of bytes) – value(s) that does NOT satisfy the condition (AND clause)
- **cond_func** – function that takes the node value and return a boolean
- **case_sensitive** – if False, ignore case for performing comparison

__module__ = 'framework.node'

_handle_cond(*val*)

check(*node*)

class framework.node.**SyncExistenceObj**(*sync_list*, *and_junction=True*)
Bases: *framework.node.SyncObj*

__init__(*sync_list*, *and_junction=True*)

__module__ = 'framework.node'

_condition_satisfied(*node*, *condition*)

check()

get_node_containers()
Shall return either a *Node* or a list of Nodes or a list of (Node, param) where param should provide __copy__ method if needed.

put_node_containers(*new_containers*)
This method will be called to provide updated containers that should replace the old ones.

Parameters **new_containers** – the updated containers

class framework.node.**SyncObj**
Bases: object

__annotations__ = {}

__module__ = 'framework.node'

_sync_nodes_specific(*src_node*)

get_node_containers()
Shall return either a *Node* or a list of Nodes or a list of (Node, param) where param should provide __copy__ method if needed.

make_private(*node_dico*)

put_node_containers(*new_containers*)
This method will be called to provide updated containers that should replace the old ones.

Parameters **new_containers** – the updated containers

synchronize_nodes(*src_node*)

class framework.node.**SyncQtyFromObj**(*node*, *base_qty=0*)
Bases: *framework.node.SyncObj*

__annotations__ = {}

__init__(*node*, *base_qty=0*)

__module__ = 'framework.node'

get_node_containers()
Shall return either a *Node* or a list of Nodes or a list of (Node, param) where param should provide __copy__ method if needed.

put_node_containers(*new_containers*)
This method will be called to provide updated containers that should replace the old ones.

Parameters **new_containers** – the updated containers

property qty

class framework.node.**SyncScope**(*value*)
Bases: enum.Enum

An enumeration.

Existence = 10

Inexistence = 11

Qty = 1

QtyFrom = 2

Size = 20

__module__ = 'framework.node'

class framework.node.**SyncSizeObj**(*node*, *base_size=0*, *apply_to_enc_size=False*)

Bases: *framework.node.SyncObj*

__annotations__ = {}

__init__(*node*, *base_size=0*, *apply_to_enc_size=False*)

__module__ = 'framework.node'

_sync_nodes_specific(*src_node*)

get_node_containers()

Shall return either a *Node* or a list of `Nodes` or a list of `(Node, param)` where `param` should provide `__copy__` method if needed.

put_node_containers(*new_containers*)

This method will be called to provide updated containers that should replace the old ones.

Parameters **new_containers** – the updated containers

set_size_on_source_node(*size*)

property size_for_absorption

framework.node.**flatten**(*nested*)

framework.node.**make_entangled_nodes**(*node_list*)

framework.node.**make_wrapped_node**(*name*, *vals=None*, *node=None*, *prefix=None*, *suffix=None*, *key_node_name='KEY_ELT'*)

framework.node.**split_verbose_with**(*predicate*, *iterable*)

framework.node.**split_with**(*predicate*, *iterable*)

13.2.5 framework.node_builder module

class framework.node_builder.**NodeBuilder**(*dm=None*, *delayed_jobs=True*, *add_env=True*, *default_gen_custo=None*, *default_nonterm_custo=None*)

Bases: `object`

HIGH_PRIO = 1

LOW_PRIO = 3

MEDIUM_PRIO = 2

RootNS = 1

VERYLOW_PRIO = 4

__get_node_from_db(*name_desc*, *namespace=None*)

__handle_clone(*desc*, *parent_node*, *namespace=None*)

__init__(*dm=None*, *delayed_jobs=True*, *add_env=True*, *default_gen_custo=None*, *default_nonterm_custo=None*)

Help the process of data description. This class is able to construct a `framework.data_model.Node` object from a JSON-like description.

Parameters

- **dm** (DataModel) – a DataModel object, only required if the 'import_from' statement is used with *create_graph_from_desc()*.
- **delayed_jobs** (*bool*) – Enable or disabled delayed jobs feature. Used for instance for delaying constraint that cannot be solved immediately.
- **add_env** (*bool*) – If *True*, an `framework.data_model.Env` object will be assigned to the generated `framework.data_model.Node` from *create_graph_from_desc()*. Should be set to `False` if you consider using the generated *Node* within another description or if you will copy it for building a new node type. Keeping an `Env()` object can be dangerous if you make some clones of it and don't pay attention to set a new `Env()` for each copy, because. A graph node SHALL have only one `Env()` shared between all the nodes and an Env() shall not be shared between independent graph (otherwise it could lead to unexpected results).
- **default_gen_custo** – override default Generator node customization
- **default_nonterm_custo** – override default NonTerminal node customization

__module__ = 'framework.node_builder'

__post_handling(*desc*, *node*, *namespace=None*)

__pre_handling(*desc*, *node*, *namespace=None*)

_clone_from_dict(*node*, *ref*, *desc*, *current_ns*)

_complete_func(*node*, *args*, *conf*, *from_ns*, *current_ns*)

_complete_generator(*node*, *args*, *conf*, *from_ns*, *current_ns*)

_complete_generator_from_desc(*node*, *args*, *conf*)

_create_generator_node(*desc*, *node=None*, *namespace=None*)

_create_graph_from_desc(*desc*, *parent_node*, *namespace=None*)

_create_leaf_node(*desc*, *node=None*, *namespace=None*)

_create_nodes_from_shape(*shapes*, *parent_node*, *shape_type='>'*, *dup_mode='u'*, *namespace=None*)

_create_non_terminal_node(*desc*, *node=None*, *namespace=None*)

_create_non_terminal_node_from_regex(*desc*, *node=None*, *namespace=None*)

_create_todo_list()

_get_from_dict(*node*, *ref*, *parent_node*)

_handle_common_attr(*node*, *desc*, *conf*, *current_ns=None*)

_handle_custo(*node*, *desc*, *conf*)

_handle_name(*name_desc*, *namespace=None*)

_register_todo(*node*, *func*, *args=None*, *unpack_args=True*, *prio=2*, *last_position=False*)

_set_env(*node*, *args*)

_set_sync_node(*node*, *comp*, *scope*, *conf*, *private*, *from_ns*)

_setup_constraints(*node*, *constraints*, *root_namespace*, *constraint_highlight*)

_update_provided_node(*desc*, *node=None*, *namespace=None*)

_verify_keys_conformity(*desc*)

create_graph_from_desc(*desc*)

ic = <framework.node.NodeInternalsCriteria object>

valid_keys = ['name', 'contents', 'qty', 'clone', 'type', 'alt', 'conf', 'custo_set', 'custo_clear', 'evolution_func', 'description', 'default_qty', 'namespace', 'from_namespace', 'highlight', 'constraints', 'constraints_highlight', 'weight', 'shape_type', 'section_type', 'duplicate_mode', 'weights', 'separator', 'prefix', 'suffix', 'unique', 'always', 'encoder', 'node_args', 'other_args', 'provide_helpers', 'trigger_last', 'specific_fuzzy_vals', 'default', 'import_from', 'data_id', 'determinist', 'random', 'finite', 'infinite', 'mutable', 'clear_attrs', 'set_attrs', 'absorb_csts', 'absorb_helper', 'semantics', 'fuzz_weight', 'sync_qty_with', 'qty_from', 'exists_if', 'exists_if_not', 'exists_if/and', 'exists_if/or', 'sync_size_with', 'sync_enc_size_with', 'post_freeze', 'charset', 'debug']

class framework.node_builder.**RegexParser**(*machine=None*)

Bases: *framework.node_builder.StateMachine*

class Brackets(*machine=None*)

Bases: *framework.node_builder.StateMachine*, *framework.node_builder.RegexParser.QtyState*

class Comma(*machine*)

Bases: *framework.node_builder.RegexParser.Brackets.Max*

INITIAL = False

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

class Final(*machine*)

Bases: *framework.node_builder.State*

INITIAL = False

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

advance(*context*)

Check transitions using the first non-run character. :param context: root state machine (global context) :type context: StateMachine

Returns Class of the next state de run (None if we are in a final state)

INITIAL = False

class Initial(*machine*)

Bases: *framework.node_builder.State*

INITIAL = True

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

advance(*ctx*)
Check transitions using the first non-run character. :param context: root state machine (global context) :type context: StateMachine
Returns Class of the next state de run (None if we are in a final state)

class Max(*machine*)
Bases: *framework.node_builder.State*

INITIAL = False

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

advance(*context*)
Check transitions using the first non-run character. :param context: root state machine (global context) :type context: StateMachine
Returns Class of the next state de run (None if we are in a final state)

class Min(*machine*)
Bases: *framework.node_builder.State*

INITIAL = False

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

advance(*context*)
Check transitions using the first non-run character. :param context: root state machine (global context) :type context: StateMachine
Returns Class of the next state de run (None if we are in a final state)

__annotations__ = {}

__module__ = 'framework.node_builder'

advance(*ctx*)
Check transitions using the first non-run character. :param context: root state machine (global context) :type context: StateMachine
Returns Class of the next state de run (None if we are in a final state)

class Choice(*machine*)
Bases: *framework.node_builder.RegexParser.Initial*

INITIAL = False

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

class Dot(*machine*)
Bases: *framework.node_builder.RegexParser.Group*

INITIAL = False

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

class Escape(*machine*)

Bases: *framework.node_builder.State*

INITIAL = False

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

advance(*ctx*)

Check transitions using the first non-run character. :param context: root state machine (global context) :type context: StateMachine

Returns Class of the next state de run (None if we are in a final state)

class EscapeMetaSequence(*machine*)

Bases: *framework.node_builder.RegexParser.Group*

INITIAL = False

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

class Final(*machine*)

Bases: *framework.node_builder.State*

INITIAL = False

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

advance(*ctx*)

Check transitions using the first non-run character. :param context: root state machine (global context) :type context: StateMachine

Returns Class of the next state de run (None if we are in a final state)

class Group(*machine*)

Bases: *framework.node_builder.State*

__annotations__ = {}

__module__ = 'framework.node_builder'

advance(*ctx*)

Check transitions using the first non-run character. :param context: root state machine (global context) :type context: StateMachine

Returns Class of the next state de run (None if we are in a final state)

class Initial(*machine*)

Bases: *framework.node_builder.State*

INITIAL = True

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

advance(*ctx*)
Check transitions using the first non-run character. :param context: root state machine (global context) :type context: StateMachine
Returns Class of the next state de run (None if we are in a final state)

class Main(*machine*)
Bases: *framework.node_builder.State*

INITIAL = False

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

advance(*ctx*)
Check transitions using the first non-run character. :param context: root state machine (global context) :type context: StateMachine
Returns Class of the next state de run (None if we are in a final state)

class Parenthesis(*machine=None*)
Bases: *framework.node_builder.StateMachine*, *framework.node_builder.RegexParser.Group*

class Choice(*machine*)
Bases: *framework.node_builder.RegexParser.Parenthesis.Initial*

INITIAL = False

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

advance(*ctx*)
Check transitions using the first non-run character. :param context: root state machine (global context) :type context: StateMachine
Returns Class of the next state de run (None if we are in a final state)

class Escape(*machine*)
Bases: *framework.node_builder.State*

INITIAL = False

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

advance(*ctx*)
Check transitions using the first non-run character. :param context: root state machine (global context) :type context: StateMachine
Returns Class of the next state de run (None if we are in a final state)

class Final(*machine*)
Bases: *framework.node_builder.State*

INITIAL = False

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*context*)

advance(*context*)
Check transitions using the first non-run character. :param context: root state machine (global context) :type context: StateMachine
Returns Class of the next state de run (None if we are in a final state)

INITIAL = False

class Initial(*machine*)
Bases: *framework.node_builder.State*

INITIAL = True

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

advance(*ctx*)
Check transitions using the first non-run character. :param context: root state machine (global context) :type context: StateMachine
Returns Class of the next state de run (None if we are in a final state)

class Main(*machine*)
Bases: *framework.node_builder.RegexParser.Parenthesis.Initial*

INITIAL = False

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

advance(*ctx*)
Check transitions using the first non-run character. :param context: root state machine (global context) :type context: StateMachine
Returns Class of the next state de run (None if we are in a final state)

__annotations__ = {}

__module__ = 'framework.node_builder'

class QtyState(*machine*)
Bases: *framework.node_builder.State*

INITIAL = False

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

advance(*ctx*)
Check transitions using the first non-run character. :param context: root state machine (global context) :type context: StateMachine
Returns Class of the next state de run (None if we are in a final state)

class SquareBrackets(*machine=None*)
Bases: *framework.node_builder.StateMachine*, *framework.node_builder.RegexParser.Group*

class AfterRange(*machine*)
Bases: *framework.node_builder.RegexParser.SquareBrackets.Initial*

INITIAL = False

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

advance(*ctx*)
Check transitions using the first non-run character. :param context: root state machine (global context) :type context: StateMachine
Returns Class of the next state de run (None if we are in a final state)

class BeforeRange(*machine*)
Bases: *framework.node_builder.RegexParser.SquareBrackets.Initial*

INITIAL = False

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

advance(*ctx*)
Check transitions using the first non-run character. :param context: root state machine (global context) :type context: StateMachine
Returns Class of the next state de run (None if we are in a final state)

class EscapeAfterRange(*machine*)
Bases: *framework.node_builder.State*

INITIAL = False

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

advance(*ctx*)
Check transitions using the first non-run character. :param context: root state machine (global context) :type context: StateMachine
Returns Class of the next state de run (None if we are in a final state)

class EscapeBeforeRange(*machine*)
Bases: *framework.node_builder.State*

INITIAL = False

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

advance(*ctx*)
Check transitions using the first non-run character. :param context: root state machine (global context) :type context: StateMachine

Returns Class of the next state de run (None if we are in a final state)

class EscapeMetaSequence(*machine*)
Bases: *framework.node_builder.RegexParser.SquareBrackets.BeforeRange*

INITIAL = False

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

class Final(*machine*)
Bases: *framework.node_builder.State*

INITIAL = False

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

advance(*ctx*)
Check transitions using the first non-run character. :param context: root state machine (global context) :type context: StateMachine
Returns Class of the next state de run (None if we are in a final state)

INITIAL = False

class Initial(*machine*)
Bases: *framework.node_builder.State*

INITIAL = True

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

advance(*ctx*)
Check transitions using the first non-run character. :param context: root state machine (global context) :type context: StateMachine
Returns Class of the next state de run (None if we are in a final state)

class Range(*machine*)
Bases: *framework.node_builder.State*

INITIAL = False

__annotations__ = {}

__module__ = 'framework.node_builder'

_run(*ctx*)

advance(*ctx*)
Check transitions using the first non-run character. :param context: root state machine (global context) :type context: StateMachine
Returns Class of the next state de run (None if we are in a final state)

__annotations__ = {}

__module__ = 'framework.node_builder'

__module__ = 'framework.node_builder'

_create_non_terminal_node()

_create_terminal_node(*name*, *type*, *values=None*, *alphabet=None*, *qty=None*)

append_to_alphabet(*alphabet*)

append_to_buffer(*str*)

append_to_contents(*content*)

property buffer

flush()

init_specific()
Can be overridden to express additional initializations

parse(*inputs*, *name*, *charset=2*)

reset()

start_new_shape()

start_new_shape_from_buffer()

class framework.node_builder.State(*machine*)
Bases: object

Represent states at the lower level

__annotations__ = {}

__init__(*machine*)

Parameters **machine** (StateMachine) – state machine where it lives (local context)

__module__ = 'framework.node_builder'

_run(*context*)

advance(*context*)
Check transitions using the first non-run character. :param context: root state machine (global context) :type context: StateMachine

Returns Class of the next state de run (None if we are in a final state)

init_specific()
Can be overridden to express additional initializations

run(*context*)
Do some actions on the current character. :param context: root state machine (global context) :type context: StateMachine

class framework.node_builder.StateMachine(*machine=None*)
Bases: *framework.node_builder.State*

Represent states that contain other states.

__annotations__ = {}

__init__(*machine=None*)

Parameters **machine** (StateMachine) – state machine where it lives (local context)

__module__ = 'framework.node_builder'

_run(*context*)

property input

run(*context*)

Do some actions on the current character. :param context: root state machine (global context) :type context: StateMachine

framework.node_builder.initial(*cls*)

framework.node_builder.register(*cls*)

13.2.6 framework.value_types module

class framework.value_types.BitField(*subfield_limits=None*, *subfield_sizes=None*, *subfield_values=None*, *subfield_val_extremums=None*, *padding=0*, *lsb_padding=True*, *show_padding=False*, *endian=1*, *determinist=True*, *subfield_descs=None*, *subfield_value_descs=None*, *defaults=None*)

Bases: framework.value_types.VT_Alt

Provide: - either @subfield_limits or @subfield_sizes - either @subfield_values or @subfield_val_extremums

__compute_total_possible_values()

the returned number correspond to the total number of values that can be returned by the BitField in determinist mode. This number does not cover all the values such a BitField should be able to generate. Refer to get_value() comments for more information.

__init__(*subfield_limits=None*, *subfield_sizes=None*, *subfield_values=None*, *subfield_val_extremums=None*, *padding=0*, *lsb_padding=True*, *show_padding=False*, *endian=1*, *determinist=True*, *subfield_descs=None*, *subfield_value_descs=None*, *defaults=None*)

__module__ = 'framework.value_types'

_check_constraints(*sf_values*)

_enable_fuzz_mode(*fuzz_magnitude=1.0*)

_enable_normal_mode()

_encode_bitfield(*val*)

_read_value_from(*blob*, *size*, *endian*, *constraints*)

Used by .do_absorb(). side effect: may change self.padding_one dictionary.

_reset_idx(*reset_idx_inuse=True*)

absorb_auto_helper(*blob*, *constraints*)

after_enabling_mode()

property bit_length

property byte_length

change_subfield(*idx*, *values=None*, *extremums=None*)

Change the constraints on a given subfield.

Parameters

- **idx** (*int*) – subfield index, from 0 (low significant subfield) to nb_subfields-1 (specific index -1 is used to choose the last subfield).

- **values** (*list*) – new values for the subfield (remove previous value list or remove previous extremums if no value list was used for this subfield)
- **extremums** (*list*) – new extremums for the subfield (remove previous extremums or remove previous value list if no extremums were used for this subfield)

property count_of_possible_values
the returned number correspond to the total number of values that can be returned by the BitField in determinist mode. This number does not cover all the values such a BitField should be able to generate. Refer to get_value() comments for more information.

do_absorb(*blob*, *constraints*, *off=0*, *size=None*)

do_cleanup_absorb()
To be called after self.do_absorb() or self.do_revert_absorb()

do_revert_absorb()
If needed should be called just after self.do_absorb().

extend(*bitfield*, *rightside=True*)

extend_left(*bitfield*)

extend_right(*bitfield*)

get_current_raw_val()

get_current_value()
Provide the current value of the object. Should not change the state of the object except if no current values.

Returns: bytes

get_subfield(*idx*)

get_value()
In determinist mode, all the values such a BitField should be able to generate are not covered but only a subset of them (i.e., all combinations are not computed). It has been chosen to only keep the value based on the following algorithm: "exhaust each subfield one at a time".

Rationale: In most cases, computing all combinations does not make sense for fuzzing purpose.

idx_from_desc(*sf_desc*)

is_compatible(*integer*, *size*)

is_exhausted()

make_determinist()

make_private(*forget_current_state*)

make_random()

padding_one = [0, 1, 3, 7, 15, 31, 63, 127]

pretty_print(*max_size=None*)

reset_state()

rewind()

set_bitfield(*sf_values=None*, *sf_val_extremums=None*, *sf_limits=None*, *sf_sizes=None*, *sf_descs=None*, *sf_val_descs=None*, *sf_defaults=None*)

set_default_value(*sf_values*)

set_size_from_constraints(*size=None*, *encoded_size=None*)

set_subfield(*idx*, *val*)

Parameters

- **idx** – Either an integer which should be the subfield index, from 0 (low significant subfield) to nb_subfields-1 (specific index -1 is used to choose the last subfield). Or a string which should be the description of the field.
- **val** (*int*) – new value for the subfield

update_raw_value(*val*)

class framework.value_types.**Filename**(*values=None*, *size=None*, *min_sz=None*, *max_sz=None*, *determinist=True*, *codec='latin-1'*, *case_sensitive=True*, *default=None*, *extra_fuzzy_list=None*, *absorb_regexp=None*, *alphabet=None*, *min_encoded_sz=None*, *max_encoded_sz=None*, *encoding_arg=None*, *values_desc=None*, ***kwargs*)

Bases: *framework.value_types.String*

__annotations__ = {}

__module__ = 'framework.value_types'

_get_path_depth(*path*)

_get_path_from_value(*value*, *knowledge*)

Returned path always terminates with a separator

linux_prefix = [b'../', b'..\xc0\xaf', b'\xc0\xae\xc0\xae\xc0\xaf']

```
linux_specific_fnames =
[b'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

b'././././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././././././././././././././
./././././././././././././././././././././././././././././././././TEST']

linux_suffix = [b'etc/password']
```

path_mode = False

subclass_specific_init(*specific_suffix=None*, *uri_parsing=False*)
Specific init for Filename

Parameters

- **specific_suffix** – List of specific suffixes that will be used for path traversal test cases in addition to the current list.
- **uri_parsing** – if the filename is to be consumed as an URI

subclass_specific_test_cases(*knowledge*, *orig_val*, *fuzz_magnitude=1.0*)
To be overwritten by class that inherits from String if specific test cases need to be implemented

Parameters

- **knowledge** –
- **orig_val** –
- **fuzz_magnitude** –

Returns list of test cases or None

Return type list

uri_prefix = [b'%2e%2e%2f', b'%2e%2e/', b'..%2f', b'..%252f', b'.%252e/', b'%2e%2e%5c', b'..%255c', b'..%c0%af', b'%c0%ae%c0%ae%c0%af']

uri_suffix = [b'MARKER.txt']

windows_prefix = [b'..\\', b'\xc0\xae\xc0\xae\\']

windows_specific_fnames = [b'PRN', b'NUL.txt', b'C:\\..\\..\\', b'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA txt']

windows_suffix = [b'Windows\\system.ini']

class framework.value_types.**FolderPath**(*values=None*, *size=None*, *min_sz=None*, *max_sz=None*, *determinist=True*, *codec='latin-1'*, *case_sensitive=True*, *default=None*, *extra_fuzzy_list=None*, *absorb_regexp=None*, *alphabet=None*, *min_encoded_sz=None*, *max_encoded_sz=None*, *encoding_arg=None*, *values_desc=None*, ***kwargs*)
Bases: *framework.value_types.Filename*

__annotations__ = {}

__module__ = 'framework.value_types'

subclass_specific_test_cases(*knowledge*, *orig_val*, *fuzz_magnitude=1.0*)
To be overwritten by class that inherits from String if specific test cases need to be implemented

Parameters

- **knowledge** –
- **orig_val** –
- **fuzz_magnitude** –

Returns list of test cases or None

Return type list

class framework.value_types.**GSM7bitPacking**(*values=None*, *size=None*, *min_sz=None*, *max_sz=None*, *determinist=True*, *codec='latin-1'*, *case_sensitive=True*, *default=None*, *extra_fuzzy_list=None*, *absorb_regexp=None*, *alphabet=None*, *min_encoded_sz=None*, *max_encoded_sz=None*, *encoding_arg=None*, *values_desc=None*, ***kwargs*)

Bases: *framework.value_types.String*

__annotations__ = {}

__module__ = 'framework.value_types'

_encoder_arg = None

_encoder_cls
alias of *framework.encoders.GSM7bitPacking_Enc*

init_encoder()

class framework.value_types.**GSMPhoneNum**(*values=None*, *size=None*, *min_sz=None*, *max_sz=None*, *determinist=True*, *codec='latin-1'*, *case_sensitive=True*, *default=None*, *extra_fuzzy_list=None*, *absorb_regexp=None*, *alphabet=None*, *min_encoded_sz=None*, *max_encoded_sz=None*, *encoding_arg=None*, *values_desc=None*, ***kwargs*)

Bases: *framework.value_types.String*

__annotations__ = {}

__module__ = 'framework.value_types'

_encoder_arg = None

_encoder_cls
alias of *framework.encoders.GSMPhoneNum_Enc*

init_encoder()

class framework.value_types.**GZIP**(*values=None*, *size=None*, *min_sz=None*, *max_sz=None*, *determinist=True*, *codec='latin-1'*, *case_sensitive=True*, *default=None*, *extra_fuzzy_list=None*, *absorb_regexp=None*, *alphabet=None*, *min_encoded_sz=None*, *max_encoded_sz=None*, *encoding_arg=None*, *values_desc=None*, ***kwargs*)

Bases: *framework.value_types.String*

__annotations__ = {}

__module__ = 'framework.value_types'

_encoder_arg = None

_encoder_cls
alias of *framework.encoders.GZIP_Enc*

init_encoder()

class framework.value_types.**INT**(*values=None*, *min=None*, *max=None*, *default=None*, *determinist=True*, *force_mode=False*, *fuzz_mode=False*, *values_desc=None*)

Bases: *framework.value_types.VT*

Base class to be inherited and not used directly

GEN_MAX_INT = 4294967296

GEN_MIN_INT = -4294967296

__annotations__ = {}

__init__(*values=None*, *min=None*, *max=None*, *default=None*, *determinist=True*, *force_mode=False*, *fuzz_mode=False*, *values_desc=None*)

__module__ = 'framework.value_types'

_check_constraints_and_update(*val*, *no_update=False*)

_convert_value(*val*)

_read_value_from(*blob*, *size*)

_unconvert_value(*val*)

absorb_auto_helper(*blob*, *constraints*)

add_specific_fuzzy_vals(*vals*)

alt_cformat = None

cformat = None

copy_attrs_from(*vt*)

determinist = True

do_absorb(*blob*, *constraints*, *off=0*, *size=None*)

do_cleanup_absorb()

do_revert_absorb()
If needed should be called just after self.do_absorb().

endian = None

fuzzy_values = None

get_current_raw_val()

get_current_value()
Provide the current value of the object. Should not change the state of the object except if no current values.

Returns: bytes

get_fuzzed_vt_list()

get_specific_fuzzy_vals()

get_value()
Walk other the values of the object on a per-call basis.

Returns: bytes

is_compatible(*integer*)

is_exhausted()

is_size_compatible(*integer*)

make_determinist()

make_private(*forget_current_state*)

make_random()

maxi = None

maxi_gen = None

mini = None

mini_gen = None

pretty_print(*max_size=None*)

reset_state()

rewind()

set_default_value(*val*)

set_size_from_constraints(*size=None*, *encoded_size=None*)

size = None

update_raw_value(*val*)

usable = False

value_space_size = None

class framework.value_types.**INT16**(*values=None*, *min=None*, *max=None*, *default=None*, *determinist=True*, *force_mode=False*, *fuzz_mode=False*, *values_desc=None*)

Bases: *framework.value_types.INT*

__annotations__ = {}

__module__ = 'framework.value_types'

fuzzy_values = [65535, 0, 32768, 32767]

size = 16

usable = False

value_space_size = 65535

class framework.value_types.**INT32**(*values=None*, *min=None*, *max=None*, *default=None*, *determinist=True*, *force_mode=False*, *fuzz_mode=False*, *values_desc=None*)

Bases: *framework.value_types.INT*

__annotations__ = {}

__module__ = 'framework.value_types'

fuzzy_values = [4294967295, 0, 2147483648, 2147483647]

size = 32

usable = False

value_space_size = 4294967295

class framework.value_types.**INT64**(*values=None*, *min=None*, *max=None*, *default=None*, *determinist=True*, *force_mode=False*, *fuzz_mode=False*, *values_desc=None*)

Bases: *framework.value_types.INT*

__annotations__ = {}

__module__ = 'framework.value_types'

fuzzy_values = [18446744073709551615, 0, 9223372036854775808, 9223372036854775807, 1229782938247303441]

size = 64

usable = False

value_space_size = 18446744073709551615

class framework.value_types.INT8(*values=None*, *min=None*, *max=None*, *default=None*, *determinist=True*, *force_mode=False*, *fuzz_mode=False*, *values_desc=None*)

Bases: *framework.value_types.INT*

__annotations__ = {}

__module__ = 'framework.value_types'

fuzzy_values = [255, 0, 1, 128, 127]

size = 8

usable = False

value_space_size = 255

class framework.value_types.INT_str(*values=None*, *min=None*, *max=None*, *default=None*, *determinist=True*, *force_mode=False*, *fuzz_mode=False*, *base=10*, *letter_case='upper'*, *min_size=None*, *reverse=False*)

Bases: *framework.value_types.INT*

__annotations__ = {}

__init__(*values=None*, *min=None*, *max=None*, *default=None*, *determinist=True*, *force_mode=False*, *fuzz_mode=False*, *base=10*, *letter_case='upper'*, *min_size=None*, *reverse=False*)

__module__ = 'framework.value_types'

_convert_value(*val*)

_prepare_format_str(*min_size*, *base*, *letter_case*)

_read_value_from(*blob*, *size*)

_unconvert_value(*val*)

copy_attrs_from(*vt*)

endian = 3

fuzzy_values = [0, -1, -4294967296, 4294967295, 4294967296]

get_fuzzed_vt_list()

is_compatible(*integer*)

pretty_print(*max_size=None*)

regex_bin = b'-?[01]'

regex_decimal = b'-?\\d'

regex_lower_hex = b'-?[0123456789abcdef]'

regex_octal = b'-?[01234567]'

regex_upper_hex = b'-?[0123456789ABCDEF]'

usable = True

value_space_size = -1

class framework.value_types.**SINT16_be**(*values=None*, *min=None*, *max=None*, *default=None*, *determinist=True*, *force_mode=False*, *fuzz_mode=False*, *values_desc=None*)

Bases: *framework.value_types.INT16*

__annotations__ = {}

__module__ = 'framework.value_types'

alt_cformat = '>H'

cformat = '>h'

endian = 1

maxi = 32767

mini = -32768

usable = True

class framework.value_types.**SINT16_le**(*values=None*, *min=None*, *max=None*, *default=None*, *determinist=True*, *force_mode=False*, *fuzz_mode=False*, *values_desc=None*)

Bases: *framework.value_types.INT16*

__annotations__ = {}

__module__ = 'framework.value_types'

alt_cformat = '<H'

cformat = '<h'

endian = 2

maxi = 32767

mini = -32768

usable = True

class framework.value_types.**SINT32_be**(*values=None*, *min=None*, *max=None*, *default=None*, *determinist=True*, *force_mode=False*, *fuzz_mode=False*, *values_desc=None*)

Bases: *framework.value_types.INT32*

__annotations__ = {}

__module__ = 'framework.value_types'

alt_cformat = '>L'

cformat = '>l'

endian = 1

maxi = 2147483647

mini = -2147483648

usable = True

class framework.value_types.**SINT32_le**(*values=None*, *min=None*, *max=None*, *default=None*, *determinist=True*, *force_mode=False*, *fuzz_mode=False*, *values_desc=None*)

Bases: *framework.value_types.INT32*

__annotations__ = {}

__module__ = 'framework.value_types'

alt_cformat = '<L'

cformat = '<l'

endian = 2

maxi = 2147483647

mini = -2147483648

usable = True

class framework.value_types.SINT64_be(*values=None*, *min=None*, *max=None*, *default=None*, *determinist=True*, *force_mode=False*, *fuzz_mode=False*, *values_desc=None*)

Bases: *framework.value_types.INT64*

__annotations__ = {}

__module__ = 'framework.value_types'

alt_cformat = '>Q'

cformat = '>q'

endian = 1

maxi = 9223372036854775807

mini = -9223372036854775808

usable = True

class framework.value_types.SINT64_le(*values=None*, *min=None*, *max=None*, *default=None*, *determinist=True*, *force_mode=False*, *fuzz_mode=False*, *values_desc=None*)

Bases: *framework.value_types.INT64*

__annotations__ = {}

__module__ = 'framework.value_types'

alt_cformat = '<Q'

cformat = '<q'

endian = 2

maxi = 9223372036854775807

mini = -9223372036854775808

usable = True

class framework.value_types.SINT8(*values=None*, *min=None*, *max=None*, *default=None*, *determinist=True*, *force_mode=False*, *fuzz_mode=False*, *values_desc=None*)

Bases: *framework.value_types.INT8*

__annotations__ = {}

__module__ = 'framework.value_types'

alt_cformat = 'B'

cformat = 'b'

endian = 3

maxi = 127

mini = -128

usable = True

class framework.value_types.**String**(*values=None*, *size=None*, *min_sz=None*, *max_sz=None*, *determinist=True*, *codec='latin-1'*, *case_sensitive=True*, *default=None*, *extra_fuzzy_list=None*, *absorb_regexp=None*, *alphabet=None*, *min_encoded_sz=None*, *max_encoded_sz=None*, *encoding_arg=None*, *values_desc=None*, ***kwargs*)

Bases: *framework.value_types.VT_Alt*

Value type that represents a character string.

encoded_string
shall be set to True by any subclass that deals with encoding

Type bool

subclass_fuzzing_list
attribute to be added by subclasses that provide specific test cases.

Type list

ASCII = 'ascii'

DEFAULT_MAX_SZ = 10000

LATIN_1 = 'iso8859-1'

UTF16BE = 'utf-16-be'

UTF16LE = 'utf-16-le'

__annotations__ = {}

__init__(*values=None*, *size=None*, *min_sz=None*, *max_sz=None*, *determinist=True*, *codec='latin-1'*, *case_sensitive=True*, *default=None*, *extra_fuzzy_list=None*, *absorb_regexp=None*, *alphabet=None*, *min_encoded_sz=None*, *max_encoded_sz=None*, *encoding_arg=None*, *values_desc=None*, ***kwargs*)

Initialize the String

Parameters

- **values** – List of the character strings that are considered valid for the node backed by this *String object*. The first item of the list is the default value
- **size** – Valid character string size for the node backed by this *String object*.
- **min_sz** – Minimum valid size for the character strings for the node backed by this *String object*. If not set, this parameter will be automatically inferred by looking at the parameter `values` whether this latter is provided.
- **max_sz** – Maximum valid size for the character strings for the node backed by this *String object*. If not set, this parameter will be automatically inferred by looking at the parameter `values` whether this latter is provided.
- **determinist** – If set to `True` generated values will be in a deterministic order, otherwise in a random order.
- **codec** – codec to use for encoding the string (e.g., 'latin-1', 'utf8')

- **case_sensitive** – If the string is set to be case sensitive then specific additional test cases will be generated in fuzzing mode.
- **default** – If not None, this value will be provided by default at first and also each time `framework.value_types.String.reset_state() is called()`.
- **extra_fuzzy_list** – During data generation, if this parameter is specified with some specific values, they will be part of the test cases generated by the generic disruptor tTYPE.
- **absorb_regexp** (*str*) – You can specify a regular expression in this parameter as a supplementary constraint for data absorption operation.
- **alphabet** – The alphabet to use for generating data, in case no `values` is provided. Also use during absorption to validate the contents. It is checked if there is no `values`.
- **values_desc** (*dict*) – Dictionary that maps string values to their descriptions (character strings). Leveraged for display purpose. Even if provided, all values do not need to be described.
- **min_encoded_sz** – Only relevant for subclasses that leverage the encoding infrastructure. Enable to provide the minimum legitimate size for an encoded string.
- **max_encoded_sz** – Only relevant for subclasses that leverage the encoding infrastructure. Enable to provide the maximum legitimate size for an encoded string.
- **encoding_arg** – Only relevant for subclasses that leverage the encoding infrastructure and that allow their encoding scheme to be configured. This parameter is directly provided to `String.init_encoding_scheme()`. Any object that go through this parameter should support the `__copy__` method.
- **kwargs** – for subclass usage

__module__ = 'framework.value_types'

__repr__()
Return repr(self).

_bytes2str(*val*)

_check_alphabet(*val*, *constraints*)

_check_constraints(*value*, *force_max_enc_sz*, *force_min_enc_sz*, *update_list=False*)

_check_constraints_and_update(*val*)

_check_contents(*val*, *val_sz*, *constraints*)

_check_size_constraints(*value*)

_enable_fuzz_mode(*fuzz_magnitude=1.0*)

_enable_normal_mode()

_encoder_cls = None

_encoder_obj = None

_ensure_enc_sizes_consistency()

_populate_values(*force_max_enc_sz=False*, *force_min_enc_sz=False*)

_read_value_from(*blob*, *constraints*)

_str2bytes(*val*)

absorb_auto_helper(*blob*, *constraints*)

ctrl_char_set = '\x00\x01\x02\x03\x04\x05\x06\x07\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x7f'

decode(*val*)
Exclusively overloaded by the decorator @from_encoder

do_absorb(*blob*, *constraints*, *off=0*, *size=None*)
Core function for absorption.

Parameters

- **blob** – binary string on which to perform absorption
- **constraints** – constraints to comply with
- **off** – absorption should start at offset *off* from blob
- **size** – if provided, *size* relates to the string to be absorbed (which can be encoded)

Returns value, off, size

do_cleanup_absorb()
To be called after self.do_absorb() or self.do_revert_absorb()

do_revert_absorb()
If needed should be called just after self.do_absorb(). (safe to recall it more than once)

encode(*val*)
Exclusively overloaded by the decorator @from_encoder

encoded_string = False

encoding_test_cases(*current_val*, *max_sz*, *min_sz*, *min_encoded__sz*, *max_encoded_sz*)
To be optionally overloaded by a subclass that deals with encoding in order to provide specific test cases on encoding scheme.

Parameters

- **current_val** – the current value (not encoded)
- **max_sz** – maximum size for a not encoded string
- **min_sz** – minimum size for a not encoded string
- **min_encoded_sz** – minimum encoded size for a string
- **max_encoded_sz** – maximum encoded size for a string

Returns the list of encoded test cases

Return type list

extended_char_set = '\x80\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0¡¢£¤¥¦§¨©ª«¬\xad®¯°±²³´µ¶·¸¹º»¼½¾¿ÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖ×ØÙÚÛÜÝÞßàáâãäåæçèéêëìíîïðñòóôõö÷øùúûüýþÿ'

static fuzz_cases_c_strings(*knowledge*, *orig_val*, *sz*, *fuzz_magnitude*)
Produces test cases relevant for C strings This method is also used by INT_str()

Parameters

- **knowledge** –
- **orig_val** –
- **sz** –
- **fuzz_magnitude** –

Returns:

static fuzz_cases_ctrl_chars(*knowledge*, *orig_val*, *sz*, *max_sz*, *codec*)

Produces test cases relevant when control characters are interpreted by the consumer This method is also used by INT_str()

Parameters

- **knowledge** –
- **orig_val** –
- **sz** –
- **max_sz** –
- **codec** –

Returns:

static fuzz_cases_letter_case(*knowledge*, *orig_val*)

Produces test cases relevant if the described element is case sensitive.

Parameters

- **knowledge** –
- **orig_val** –

Returns:

get_current_raw_val(*str_form=False*)

get_current_value()

Provide the current value of the object. Should not change the state of the object except if no current values.

Returns: bytes

get_value()

Walk other the values of the object on a per-call basis.

Returns: bytes

init_encoder = None

is_exhausted()

make_determinist()

make_private(*forget_current_state*)

make_random()

**non_ctrl_char = ' !"#$%&\'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\
]^_`abcdefghijklmnopqrstuvwxyz{|}~\x80\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\
x8c\x8d\x8e\x8f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\
xa0¡¢£¤¥¦§¨©ª«¬\
xad®¯°±²³´µ¶·¸¹º»¼½¾¿ÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖ×ØÙÚÛÜÝÞßàáâãäåæçèéêëìíîïðñòóôõö÷øùúûüýþÿ'**

pretty_print(*max_size=None*)

**printable_char_set = ' !"#$%&\'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\
\]^_`abcdefghijklmnopqrstuvwxyz{|}~'**

reset_encoder()

reset_state()

rewind()

set_default_value(*val*)

set_description(*values=None*, *size=None*, *min_sz=None*, *max_sz=None*, *determinist=True*, *codec='latin-1'*, *case_sensitive=True*, *default=None*, *extra_fuzzy_list=None*, *absorb_regexp=None*, *alphabet=None*, *min_encoded_sz=None*, *max_encoded_sz=None*)
@size take precedence over @min_sz and @max_sz

set_size_from_constraints(*size=None*, *encoded_size=None*)

subclass_specific_init(***kwargs*)
To be overwritten by class that inherits from String if specific init is necessary, for instance new parameters.

Parameters ****kwargs** –

Returns:

subclass_specific_test_cases(*knowledge*, *orig_val*, *fuzz_magnitude*)
To be overwritten by class that inherits from String if specific test cases need to be implemented

Parameters

- **knowledge** –
- **orig_val** –
- **fuzz_magnitude** –

Returns list of test cases or None

Return type list

class framework.value_types.**UINT16_be**(*values=None*, *min=None*, *max=None*, *default=None*, *determinist=True*, *force_mode=False*, *fuzz_mode=False*, *values_desc=None*)

Bases: *framework.value_types.INT16*

__annotations__ = {}

__module__ = 'framework.value_types'

alt_cformat = '>h'

cformat = '>H'

endian = 1

maxi = 65535

mini = 0

usable = True

class framework.value_types.**UINT16_le**(*values=None*, *min=None*, *max=None*, *default=None*, *determinist=True*, *force_mode=False*, *fuzz_mode=False*, *values_desc=None*)

Bases: *framework.value_types.INT16*

__annotations__ = {}

__module__ = 'framework.value_types'

alt_cformat = '<h'

cformat = '<H'

endian = 2

maxi = 65535

mini = 0

usable = True

class framework.value_types.**UINT32_be**(*values=None*, *min=None*, *max=None*, *default=None*, *determinist=True*, *force_mode=False*, *fuzz_mode=False*, *values_desc=None*)

Bases: *framework.value_types.INT32*

__annotations__ = {}

__module__ = 'framework.value_types'

alt_cformat = '>l'

cformat = '>L'

endian = 1

maxi = 4294967295

mini = 0

usable = True

class framework.value_types.**UINT32_le**(*values=None*, *min=None*, *max=None*, *default=None*, *determinist=True*, *force_mode=False*, *fuzz_mode=False*, *values_desc=None*)

Bases: *framework.value_types.INT32*

__annotations__ = {}

__module__ = 'framework.value_types'

alt_cformat = '<l'

cformat = '<L'

endian = 2

maxi = 4294967295

mini = 0

usable = True

class framework.value_types.**UINT64_be**(*values=None*, *min=None*, *max=None*, *default=None*, *determinist=True*, *force_mode=False*, *fuzz_mode=False*, *values_desc=None*)

Bases: *framework.value_types.INT64*

__annotations__ = {}

__module__ = 'framework.value_types'

alt_cformat = '>q'

cformat = '>Q'

endian = 1

maxi = 18446744073709551615

mini = 0

usable = True

class framework.value_types.**UINT64_le**(*values=None*, *min=None*, *max=None*, *default=None*, *determinist=True*, *force_mode=False*, *fuzz_mode=False*, *values_desc=None*)

Bases: *framework.value_types.INT64*

__annotations__ = {}

__module__ = 'framework.value_types'

alt_cformat = '<q'

cformat = '<Q'

endian = 2

maxi = 18446744073709551615

mini = 0

usable = True

class framework.value_types.**UINT8**(*values=None*, *min=None*, *max=None*, *default=None*, *determinist=True*, *force_mode=False*, *fuzz_mode=False*, *values_desc=None*)

Bases: *framework.value_types.INT8*

__annotations__ = {}

__module__ = 'framework.value_types'

alt_cformat = 'b'

cformat = 'B'

endian = 3

maxi = 255

mini = 0

usable = True

class framework.value_types.**VT**

Bases: object

Base class to implement Types that are leveraged by typed nodes

BigEndian = 1

LittleEndian = 2

Native = 3

__annotations__ = {}

__module__ = 'framework.value_types'

add_specific_fuzzy_vals(*vals*)

copy_attrs_from(*vt*)

enc2struct = {1: '>', 2: '<', 3: '='}

endian = None

get_current_raw_val()

get_current_value()
Provide the current value of the object. Should not change the state of the object except if no current values.

Returns: bytes

get_fuzzed_vt_list()

get_specific_fuzzy_vals()

get_value()
Walk other the values of the object on a per-call basis.

Returns: bytes

is_exhausted()

knowledge_source = None

make_determinist()

make_private(*forget_current_state*)

make_random()

maxi = None

mini = None

pretty_print(*max_size=None*)

reset_state()

rewind()

set_default_value(*val*)

set_size_from_constraints(*size=None*, *encoded_size=None*)

class framework.value_types.**VT_Alt**
Bases: *framework.value_types.VT*

__annotations__ = {}

__init__()

__module__ = 'framework.value_types'

_enable_fuzz_mode(*fuzz_magnitude=1.0*)

_enable_normal_mode()

add_specific_fuzzy_vals(*vals*)

after_enabling_mode()

enable_fuzz_mode(*fuzz_magnitude=1.0*)

enable_normal_mode()

get_specific_fuzzy_vals()

switch_mode()

class framework.value_types.**Wrapper**(*values=None*, *size=None*, *min_sz=None*, *max_sz=None*, *determinist=True*, *codec='latin-1'*, *case_sensitive=True*, *default=None*, *extra_fuzzy_list=None*, *absorb_regexp=None*, *alphabet=None*, *min_encoded_sz=None*, *max_encoded_sz=None*, *encoding_arg=None*, *values_desc=None*, ***kwargs*)
Bases: *framework.value_types.String*

__annotations__ = {}

__module__ = 'framework.value_types'

_encoder_arg = None

_encoder_cls
alias of *framework.encoders.Wrap_Enc*

init_encoder()

framework.value_types.**from_encoder**(*encoder_cls*, *encoding_arg=None*)

13.2.7 framework.generic_data_makers module

class framework.generic_data_makers.**d_add_data**
Bases: *framework.tactics_helpers.Disruptor*

Add some data within the retrieved input.

__module__ = 'framework.generic_data_makers'

_args_desc = {'after': ('If True, the addition will be done after the selected node. Otherwise, it will be done before.', True, <class 'bool'>), 'atom': ('Name of the atom to add within the retrieved input. It is mutually exclusive with @raw', None, <class 'str'>), 'name': ('If provided, the added node will have this name.', None, <class 'str'>), 'path': ('Graph path to select the node on which the disruptor should apply.', None, <class 'str'>), 'raw': ('Raw value to add within the retrieved input. It is mutually exclusive with @atom.', b'', (<class 'bytes'>, <class 'str'>))}

_modelwalker_user = False

disrupt_data(*dm*, *target*, *prev_data*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.generic_data_makers.**d_call_external_program**
Bases: *framework.tactics_helpers.Disruptor*

Call an external program to deal with the data.

__annotations__ = {}

__module__ = 'framework.generic_data_makers'

_args_desc = {'cmd': ('The external command the execute.', None, (<class 'list'>, <class 'tuple'>, <class 'str'>)), 'file_mode': ('If True the data will be provided through a file to the external program, otherwise it will be provided on the command line directly.', True, <class 'bool'>), 'path': ('Graph path regexp to select nodes on which the disruptor should apply.', None, <class 'str'>)}

_get_cmd()

_modelwalker_user = False

disrupt_data(*dm*, *target*, *prev_data*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.generic_data_makers.**d_call_function**
Bases: *framework.tactics_helpers.Disruptor*

Call the function provided with the first parameter being the Data() object received as input of this disruptor, and optionally with additional parameters if @params is set. The function should return a Data() object.

The signature of the function should be compatible with:

*func(data, *args) –> Data()*

__annotations__ = {}

__module__ = 'framework.generic_data_makers'

_args_desc = {'func': ('The function that will be called with a node as its first parameter, and provided optionnaly with addtionnal parameters if @params is set.', <function <lambda>>, <class 'method'>), 'params': ('Tuple of parameters that will be provided to the function.', None, <class 'tuple'>)}

_modelwalker_user = False

disrupt_data(*dm*, *target*, *prev_data*)

class framework.generic_data_makers.**d_corrupt_bits_by_position**
Bases: *framework.tactics_helpers.Disruptor*

Corrupt bit at a specific byte.

__annotations__ = {}

__module__ = 'framework.generic_data_makers'

_args_desc = {'ascii': ('Enforce all outputs to be ascii 7bits.', False, <class 'bool'>), 'idx': ('Byte index to be corrupted (from 1 to data length).', 1, <class 'int'>), 'new_val': ('If provided change the selected byte with the new one.', None, <class 'bytes'>)}

_modelwalker_user = False

disrupt_data(*dm*, *target*, *prev_data*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.generic_data_makers.**d_corrupt_node_bits**
Bases: *framework.tactics_helpers.Disruptor*

Corrupt bits on some nodes of the data model.

__annotations__ = {}

__module__ = 'framework.generic_data_makers'

_args_desc = {'ascii': ('Enforce all outputs to be ascii 7bits.', False, <class 'bool'>), 'nb': ('Apply corruption on @nb Nodes fetched randomly within the data model.', 2, <class 'int'>), 'new_val': ('If provided change the selected byte with the new one.', None, <class 'str'>), 'path': ('Graph path regexp to select nodes on which the disruptor should apply.', None, <class 'str'>)}

_modelwalker_user = False

disrupt_data(*dm*, *target*, *prev_data*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.generic_data_makers.**d_fix_constraints**
Bases: *framework.tactics_helpers.Disruptor*

Fix data constraints.

Release constraints from input data or from only a piece of it (if the parameter *path* is provided), then recompute them. By constraints we mean every generator (or function) nodes that may embeds constraints between nodes, and every node *existence conditions*.

__annotations__ = {}

__module__ = 'framework.generic_data_makers'

_args_desc = {'clone_node': ('If True the dmaker will always return a copy of the node. (For stateless disruptors dealing with big data it can be useful to it to False.)', False, <class 'bool'>), 'path': ('Graph path regexp to select nodes on which the disruptor should apply.', None, <class 'str'>)}

_modelwalker_user = False

disrupt_data(*dm*, *target*, *prev_data*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.generic_data_makers.**d_fuzz_model_structure**
Bases: *framework.tactics_helpers.Disruptor*

Disrupt the data model structure (replace ordered sections by unordered ones).

__annotations__ = {}

__module__ = 'framework.generic_data_makers'

_args_desc = {'path': ('Graph path regexp to select nodes on which the disruptor should apply.', None, <class 'str'>)}

_modelwalker_user = False

disrupt_data(*dm*, *target*, *prev_data*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.generic_data_makers.**d_max_size**
Bases: *framework.tactics_helpers.Disruptor*

Truncate the data (or part of the data) to the provided size.

__annotations__ = {}

__module__ = 'framework.generic_data_makers'

_args_desc = {'path': ('Graph path regexp to select nodes on which the disruptor should apply.', None, <class 'str'>), 'sz': ('Truncate the data (or part of the data) to the provided size.', 10, <class 'int'>)}

_modelwalker_user = False

disrupt_data(*dm*, *target*, *prev_data*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.generic_data_makers.**d_modify_nodes**
Bases: *framework.tactics_helpers.Disruptor*

Perform modifications on the provided data. Two ways are possible:

- Either the change is performed on the content of the nodes specified by the *path* parameter with the new *value* provided, and the optional constraints for the absorption (use *node absorption* infrastructure);
- Or the changed is performed based on a dictionary provided through the parameter *multi_mod*

__annotations__ = {}

__module__ = 'framework.generic_data_makers'

_add_info(*prev_data*, *n*, *new_value*, *status*, *size*)

_args_desc = {'clone_node': ('If True the dmaker will always return a copy of the node. (For stateless disruptors dealing with big data it can be useful to set it to False.)', False, <class 'bool'>), 'constraints': ('Constraints for the absorption of the new value.', AbsNoCsts(), <class 'framework.global_resources.AbsCsts'>), 'multi_mod': ('Dictionary of <path>:<item> pairs or <NodeSemanticsCriteria>:<item> pairs or <NodeInternalsCriteria>:<item> pairs to change multiple nodes with different values. <item> can be either only the new <value> or a tuple (<value>,<abscsts>) if new constraint for absorption is needed', None, <class 'dict'>), 'path': ('Graph path regexp to select nodes on which the disruptor should apply.', None, <class 'str'>), 'sem': ('Semantics to select nodes on which the disruptor should apply.', None, (<class 'str'>, <class 'list'>)), 'unfold': ('Resolve all the generator nodes within the input before performing the @path/@sem research', False, <class 'bool'>), 'value': ('The new value to inject within the data.', b'', <class 'bytes'>)}

_modelwalker_user = False

disrupt_data(*dm*, *target*, *prev_data*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.generic_data_makers.**d_next_node_content**
Bases: *framework.tactics_helpers.Disruptor*

Move to the next content of the nodes from input data or from only a piece of it (if the parameter *path* is provided). Basically, unfreeze the nodes then freeze them again, which will consequently produce a new data.

__annotations__ = {}

__module__ = 'framework.generic_data_makers'

_args_desc = {'clone_node': ('If True the dmaker will always return a copy of the node. (for stateless disruptors dealing with big data it can be useful to it to False).', False, <class 'bool'>), 'path': ('Graph path regexp to select nodes on which the disruptor should apply.', None, <class 'str'>), 'recursive': ('Apply the disruptor recursively.', True, <class 'str'>)}

_modelwalker_user = False

disrupt_data(*dm*, *target*, *prev_data*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.generic_data_makers.**d_operate_on_nodes**
Bases: *framework.tactics_helpers.Disruptor*

Perform an operation on the nodes specified by the regexp path. @op is an operation that applies to a node and @params are a tuple containing the parameters that will be provided to @op. If no path is provided, the root node will be used.

__annotations__ = {}

__module__ = 'framework.generic_data_makers'

_add_info(*prev_data*, *n*)

_args_desc = {'clone_node': ('If True the dmaker will always return a copy of the node. (For stateless disruptors dealing with big data it can be useful to set it to False.)', False, <class 'bool'>), 'op': ('The operation to perform on the selected nodes.', <function Node.clear_attr>, <class 'method'>), 'op_ref': ("Predefined operation that can be referenced by name. The current predefined function are: 'unfreeze', 'freeze', 'walk', 'set_qty'. Take precedence over @op if not None.", None, <class 'str'>), 'params': ('Tuple of parameters that will be provided to the operation.', (), <class 'tuple'>), 'path': ('Graph path regexp to select nodes on which the disruptor should apply.', None, <class 'str'>), 'sem': ('Semantics to select nodes on which the disruptor should apply.', None, (<class 'str'>, <class 'list'>))}

_modelwalker_user = False

disrupt_data(*dm*, *target*, *prev_data*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.generic_data_makers.d_shallow_copy
Bases: *framework.tactics_helpers.Disruptor*

Shallow copy of the input data, which means: ignore its frozen state during the copy.

__annotations__ = {}

__module__ = 'framework.generic_data_makers'

_args_desc = {}

_modelwalker_user = False

disrupt_data(*dm*, *target*, *prev_data*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.generic_data_makers.d_switch_to_alternate_conf
Bases: *framework.tactics_helpers.Disruptor*

Switch to an alternate configuration.

__annotations__ = {}

__module__ = 'framework.generic_data_makers'

_args_desc = {'conf': ('Change the configuration, with the one provided (by name), of all subnodes fetched by @path, one-by-one. [default value is set dynamically with the first-found existing alternate configuration]', None, <class 'str'>), 'path': ('Graph path regexp to select nodes on which the disruptor should apply.', None, <class 'str'>), 'recursive': ('Does the reachable nodes from the selected ones need also to be changed?', True, <class 'bool'>)}

_modelwalker_user = False

disrupt_data(*dm*, *target*, *prev_data*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.generic_data_makers.**g_generic_pattern**
Bases: *framework.tactics_helpers.Generator*

Generate basic data based on a pattern and different parameters.

__annotations__ = {}

__module__ = 'framework.generic_data_makers'

_args_desc = {'eval': ('The pattern will be evaluated before being used. Note that the evaluation shall result in a byte string.', False, <class 'bool'>), 'pattern': ('Pattern to be used for generating data', b'1234567890', <class 'bytes'>), 'prefix': ('Prefix added to the pattern', b'', <class 'bytes'>), 'size': ('Size of the generated data.', None, <class 'int'>), 'suffix': ('Suffix replacing the end of the pattern', b'', <class 'bytes'>)}

_modelwalker_user = False

generate_data(*dm*, *monitor*, *target*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.generic_data_makers.**g_population**
Bases: *framework.tactics_helpers.Generator*

Walk through the given population

__annotations__ = {}

__module__ = 'framework.generic_data_makers'

_args_desc = {'population': ('The population to iterate over.', None, <class 'framework.evolutionary_helpers.Population'>), 'track': ('Keep trace of the changes that occurred on data, generation after generation', False, <class 'bool'>)}

_modelwalker_user = False

generate_data(*dm*, *monitor*, *target*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.generic_data_makers.**sd_constraint_fuzz**
Bases: *framework.tactics_helpers.StatefulDisruptor*

When the CSP (Constraint Satisfiability Problem) backend are used in the node description. This operator negates the constraint one-by-one and output 1 or more samples for each negated constraint.

__annotations__ = {}

__module__ = 'framework.generic_data_makers'

_args_desc = {'clone_node': ('If True, this operator will always return a copy of the node. (for stateless diruptors dealing with big data it can be usefull to set it to False)', True, <class 'bool'>), 'const_idx': ('Index of the constraint to begin with (first index is 1)', 1, <class 'int'>), 'sample_idx': ('Index of the sample for the selected constraint to begin with (first index is 1)', 1, <class 'int'>), 'samples_per_cst': ('Maximum number of samples to output for each negated constraint (-1 means until the end)', -1, <class 'int'>)}

_modelwalker_user = False

_process_next_constraint()

_update_csp()

disrupt_data(*dm*, *target*, *data*)
@data: it is either equal to prev_data the first time disrupt_data() is called by the FMK, or it is a an empty data (that is Data()).

set_seed(*prev_data*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.generic_data_makers.sd_fuzz_separator_nodes
Bases: *framework.tactics_helpers.StatefulDisruptor*

Perform alterations on separators (one at a time). Each time a separator is encountered in the provided data, it will be replaced by another separator picked from the ones existing within the provided data.

__annotations__ = {}

__module__ = 'framework.generic_data_makers'

_args_desc = {'clone_node': ('If True, this operator will always return a copy of the node. (for stateless diruptors dealing with big data it can be usefull to set it to False)', True, <class 'bool'>), 'deep': ('When set to True, if a node structure has changed, the modelwalker will reset its walk through the children nodes.', True, <class 'bool'>), 'init': ('Make the model walker ignore all the steps until the provided one', 1, <class 'int'>), 'max_node_tc': ('Maximum number of test cases per node (-1 means until the end). This value is used for nodes with a fuzz weight strictly greater than 1.', -1, <class 'int'>), 'max_steps': ('Maximum number of steps (-1 means until the end)', -1, <class 'int'>), 'min_node_tc': ('Minimum number of test cases per node (-1 means until the end)', -1, <class 'int'>), 'order': ('When set to True, the fuzzing order is strictly guided by the data structure. Otherwise, fuzz weight (if specified in the data model) is used for ordering.', True, <class 'bool'>), 'path': ('Graph path regexp to select nodes on which the disruptor should apply.', None, <class 'str'>), 'sem': ('Semantics to select nodes on which the disruptor should apply.', None, (<class 'str'>, <class 'list'>))}

_modelwalker_user = True

disrupt_data(*dm*, *target*, *data*)
@data: it is either equal to prev_data the first time disrupt_data() is called by the FMK, or it is a an empty data (that is Data()).

set_seed(*prev_data*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.generic_data_makers.sd_fuzz_typed_nodes
Bases: *framework.tactics_helpers.StatefulDisruptor*

Perform alterations on typed nodes (one at a time) according to: - their type (e.g., INT, Strings, ...) - their attributes (e.g., allowed values, minimum size, ...) - knowledge retrieved from the data (e.g., if the input data uses separators, their symbols are leveraged in the fuzzing) - knowledge on the target retrieved from the project file or dynamically from feedback inspection (e.g., C language, GNU/Linux OS, ...)

If the input has different shapes (described in non-terminal nodes), this will be taken into account by fuzzing every shape combinations.

Note: this disruptor includes what tSEP does and goes beyond with respect to separators.

__annotations__ = {}

__module__ = 'framework.generic_data_makers'

_args_desc = {'clone_node': ('If True, this operator will always return a copy of the node. (for stateless diruptors dealing with big data it can be usefull to set it to False)', True, <class 'bool'>), 'consider_sibbling_change': ('[EXPERIMENTAL] While walking through terminal nodes, if sibbling nodes are no more the same because of existence condition for instance, walk through the new nodes. (Currently, work only with some specific data model construction.)', False, <class 'bool'>), 'deep': ('When set to True, if a node structure has changed, the modelwalker will reset its walk through the children nodes.', True, <class 'bool'>), 'fix': ("Limit constraints fixing to the nodes related to the currently fuzzed one (only implemented for 'sync_size_with' and 'sync_enc_size_with').", True, <class 'bool'>), 'fix_all': ('For each produced data, reevaluate the constraints on the whole graph.', False, <class 'bool'>), 'full_combinatory': ('When set to True, enable full-combinatory mode for non-terminal nodes. It means that the non-terminal nodes will be customized in "FullCombinatory" mode', False, <class 'bool'>), 'fuzz_mag': ('Order of magnitude for maximum size of some fuzzing test cases.', 1.0, <class 'float'>), 'ign_mutable_attr': ('Walk through all the nodes even if their Mutable attribute is cleared.', False, <class 'bool'>), 'ign_sep': ('When set to True, separators will be ignored if any are defined.', False, <class 'bool'>), 'init': ('Make the model walker ignore all the steps until the provided one', 1, <class 'int'>), 'leaf_determinism': ("If set to 'True', all the typed nodes of the model will be set to determinist mode prior to any fuzzing. If set to 'False', they will be set to random mode. Otherwise, if set to 'None', nothing will be done.", None, <class 'bool'>), 'leaf_fuzz_determinism': ("If set to 'True', each typed node will be fuzzed in a deterministic way. If set to 'False' each typed node will be fuzzed in a random way. Otherwise, if it is set to 'None', it will be guided by the data model determinism. Note: this option is complementary to 'determinism' as it acts on the typed node substitutions that occur through this disruptor", True, <class 'bool'>), 'make_determinist': ("If set to 'True', the whole model will be set in determinist mode.Otherwise it will be guided by the data model determinism.", False, <class 'bool'>), 'max_node_tc': ('Maximum number of test cases per node (-1 means until the end). This value is used for nodes with a fuzz weight strictly greater than 1.', -1, <class 'int'>), 'max_steps': ('Maximum number of steps (-1 means until the end)', -1, <class 'int'>), 'min_node_tc': ('Minimum number of test cases per node (-1 means until the end)', -1, <class 'int'>), 'order': ('When set to True, the fuzzing order is strictly guided by the data structure. Otherwise, fuzz weight (if specified in the data model) is used for ordering.', True, <class 'bool'>), 'path': ('Graph path regexp to select nodes on which the disruptor should apply.', None, <class 'str'>), 'sem': ('Semantics to select nodes on which the disruptor should apply.', None, (<class 'str'>, <class 'list'>))}

_modelwalker_user = True

disrupt_data(*dm*, *target*, *data*)

@data: it is either equal to prev_data the first time disrupt_data() is called by the FMK, or it is a an empty data (that is Data()).

set_seed(*prev_data*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.generic_data_makers.**sd_struct_constraints**
Bases: *framework.tactics_helpers.StatefulDisruptor*

Perform constraints alteration (one at a time) on each node that depends on another one regarding its existence, its quantity, its size, ...

If *deep* is set, enable more corruption cases on the data structure, based on the internals of each non-terminal node: - the minimum and maximum amount of the subnodes of each non-terminal nodes - ...

__annotations__ = {}

__module__ = 'framework.generic_data_makers'

_args_desc = {'deep': ('If True, enable corruption of non-terminal node internals', False, <class 'bool'>), 'init': ('Make the model walker ignore all the steps until the provided one.', 1, <class 'int'>), 'max_steps': ('Maximum number of steps (-1 means until the end).', -1, <class 'int'>), 'path': ('Graph path regexp to select nodes on which the disruptor should apply.', None, <class 'str'>), 'sem': ('Semantics to select nodes on which the disruptor should apply.', None, (<class 'str'>, <class 'list'>))}

_modelwalker_user = False

disrupt_data(*dm*, *target*, *data*)
@data: it is either equal to prev_data the first time disrupt_data() is called by the FMK, or it is a an empty data (that is Data()).

set_seed(*prev_data*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.generic_data_makers.**sd_switch_to_alternate_conf**
Bases: *framework.tactics_helpers.StatefulDisruptor*

Switch the configuration of each node, one by one, with the provided alternate configuration.

__annotations__ = {}

__module__ = 'framework.generic_data_makers'

_args_desc = {'clone_node': ('If True, this operator will always return a copy of the node. (for stateless diruptors dealing with big data it can be usefull to set it to False)', True, <class 'bool'>), 'conf': ('Change the configuration, with the one provided (by name), of all nodes reachable from the root, one-by-one. [default value is set dynamically with the first-found existing alternate configuration]', None, (<class 'str'>, <class 'list'>, <class 'tuple'>)), 'init': ('Make the model walker ignore all the steps until the provided one', 1, <class 'int'>), 'max_node_tc': ('Maximum number of test cases per node (-1 means until the end). This value is used for nodes with a fuzz weight strictly greater than 1.', -1, <class 'int'>), 'max_steps': ('Maximum number of steps (-1 means until the end)', -1, <class 'int'>), 'min_node_tc': ('Minimum number of test cases per node (-1 means until the end)', -1, <class 'int'>)}

_modelwalker_user = True

disrupt_data(*dm*, *target*, *data*)
@data: it is either equal to prev_data the first time disrupt_data() is called by the FMK, or it is a an empty data (that is Data()).

set_seed(*prev_data*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.generic_data_makers.**sd_walk_csp_solutions**
Bases: *framework.tactics_helpers.StatefulDisruptor*

When the CSP (Constraint Satisfiability Problem) backend are used in the data description. This operator walk through the solutions of the CSP.

__annotations__ = {}

__module__ = 'framework.generic_data_makers'

_args_desc = {'clone_node': ('If True, this operator will always return a copy of the node. (for stateless diruptors dealing with big data it can be usefull to set it to False)', True, <class 'bool'>), 'init': ('Make the operator ignore all the steps until the provided one', 1, <class 'int'>), 'notify_exhaustion': ('When all the solutions of the CSP have been walked through, the disruptor will notify it if this parameter is set to True.', True, <class 'bool'>)}

_modelwalker_user = False

disrupt_data(*dm*, *target*, *data*)
@data: it is either equal to prev_data the first time disrupt_data() is called by the FMK, or it is a an empty data (that is Data()).

set_seed(*prev_data*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.generic_data_makers.**sd_walk_data_model**
Bases: *framework.tactics_helpers.StatefulDisruptor*

Walk through the provided data and for each visited node, iterates over the allowed values (with respect to the data model). Note: *no alteration* is performed by this disruptor.

__annotations__ = {}

__module__ = 'framework.generic_data_makers'

_args_desc = {'clone_node': ('If True, this operator will always return a copy of the node. (for stateless diruptors dealing with big data it can be usefull to set it to False)', True, <class 'bool'>), 'consider_sibbling_change': ('While walking through terminal nodes, if sibbling nodes are no more the same because of existence condition for instance, walk through the new nodes.', True, <class 'bool'>), 'deep': ('When set to True, if a node structure has changed, the modelwalker will reset its walk through the children nodes.', True, <class 'bool'>), 'fix_all': ('For each produced data, reevaluate the constraints on the whole graph.', True, <class 'bool'>), 'full_combinatory': ('When set to True, enable full-combinatory mode for non-terminal nodes. It means that the non-terminal nodes will be customized in "FullCombinatory" mode', False, <class 'bool'>), 'ign_mutable_attr': ('Walk through all the nodes even if their Mutable attribute is cleared.', True, <class 'bool'>), 'init': ('Make the model walker ignore all the steps until the provided one', 1, <class 'int'>), 'leaf_determinism': ("If set to 'True', all the typed nodes of the model will be set to determinist mode prior to any fuzzing. If set to 'False', they will be set to random mode. Otherwise, if set to 'None', nothing will be done.", None, <class 'bool'>), 'max_node_tc': ('Maximum number of test cases per node (-1 means until the end). This value is used for nodes with a fuzz weight strictly greater than 1.', -1, <class 'int'>), 'max_steps': ('Maximum number of steps (-1 means until the end)', -1, <class 'int'>), 'min_node_tc': ('Minimum number of test cases per node (-1 means until the end)', -1, <class 'int'>), 'nt_only': ('Walk through non-terminal nodes only.', False, <class 'bool'>), 'order': ('When set to True, the walking order is strictly guided by the data structure. Otherwise, fuzz weight (if specified in the data model) is used for ordering.', True, <class 'bool'>), 'path': ('Graph path regexp to select nodes on which the disruptor should apply.', None, <class 'str'>), 'sem': ('Semantics to select nodes on which the disruptor should apply.', None, (<class 'str'>, <class 'list'>))}

_modelwalker_user = True

disrupt_data(*dm*, *target*, *data*)
@data: it is either equal to prev_data the first time disrupt_data() is called by the FMK, or it is a an empty data (that is Data()).

set_seed(*prev_data*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

framework.generic_data_makers.**truncate_info**(*info*, *max_size=60*)

13.2.8 framework.target_helpers module

class framework.target_helpers.**EmptyTarget**(*verbose=False*)
Bases: *framework.target_helpers.Target*

__init__(*verbose=False*)

__module__ = 'framework.target_helpers'

_feedback_mode = 1

send_data(*data*, *from_fmk=False*)
To be overloaded.

Note: use data.to_bytes() to get binary data.

Parameters

- **from_fmk** (*bool*) – set to True if the call was performed by the framework itself, otherwise the call comes from user-code (e.g., from a *probe* or an *operator*)
- **data** (Data) – data container that embeds generally a modeled data accessible through *data.content*. However if the latter is None, it only embeds the raw data.

send_multiple_data(*data_list*, *from_fmk=False*)

Used to send multiple data to the target, or to stimulate several target's inputs in one shot.

Note: Use data.to_bytes() to get binary data

Parameters

- **from_fmk** (*bool*) – set to True if the call was performed by the framework itself, otherwise the call comes from user-code (e.g., from a *Probe* or an *Operator*)
- **data_list** (*list*) – list of data to be sent

supported_feedback_mode = [1, 2]

class framework.target_helpers.**Target**(*name=None*, *display_feedback=True*)

Bases: object

Class abstracting the real target we interact with.

About feedback: Feedback retrieved from a real target has to be provided to the user (i.e., the framework) through either after Target.send_data() is called or when Target.collect_unsolicited_feedback() is called.

FBK_WAIT_FULL_TIME = 1

FBK_WAIT_UNTIL_RECV = 2

STATUS_THRESHOLD_FOR_RECOVERY = 0

__annotations__ = {}

__init__(*name=None*, *display_feedback=True*)

__module__ = 'framework.target_helpers'

__str__()

Return str(self).

_altered_data_queued = None

_extensions = None

_feedback_mode = None

_last_sending_date = None

_logger = None

_pending_data = None

_pending_data_id = None

_send_data_lock = <unlocked _thread.lock object>

_set_feedback_timeout_specific(*fbk_timeout*)

Overload this function to handle feedback specifics

Parameters **fbk_timeout** (*float*) – time duration for collecting the feedback

_start(*target_desc*, *tg_id*)

_started = None

_stop(*target_desc*, *tg_id*)

add_extensions(*probe*)

add_pending_data(*data*)

cleanup()
To be overloaded if something needs to be performed after each data emission. It is called after any feedback has been retrieved.

collect_unsolicited_feedback(*timeout=0*)
If overloaded, it should collect any data from the associated real target that may be sent without solicitation (i.e. without any data sent through it) and make it available through the method .get_feedback()

Parameters **timeout** – Maximum delay before returning from feedback collecting

Returns False if it is not possible, otherwise it should be True

Return type bool

del_extensions()

display_feedback = False

property extensions

property fbk_wait_full_time_slot_mode

fbk_wait_full_time_slot_msg = 'Wait for the full time slot allocated for feedback retrieval'

property fbk_wait_until_recv_mode

fbk_wait_until_recv_msg = 'Wait until the target has sent something back to us'

feedback_timeout = None

get_description()

static get_fbk_mode_desc(*fbk_mode*, *short=False*)

get_feedback()
If overloaded, should return a FeedbackCollector object.

get_last_target_ack_date()
If different from None the return value is used by the FMK to log the date of the target acknowledgment after a message has been sent to it.

[Note: If this method is overloaded, is_feedback_received() should also be]

is_feedback_received()
To be overloaded if the target implements FBK_WAIT_UNTIL_RECV mode, so that it can informs the framework about feedback reception.

is_processed_data_altered()

is_started()

is_target_ready_for_new_data()
To be overloaded if the target needs some time (for conditions to occur) before data can be sent. Note: The FMK busy wait on this method() before sending a new data.

name = None

record_info(*info*)
Can be used by the target to record some information during initialization or anytime it make sense for your purpose.

Parameters **info** (*str*) – info to be recorded

Returns None

recover_target()
Implementation of target recovering operations, when a target problem has been detected (i.e. a negative feedback from a probe, an operator or the Target() itself)

Returns True if the target has been recovered. False otherwise.

Return type bool

send_data(*data*, *from_fmk=False*)
To be overloaded.

Note: use data.to_bytes() to get binary data.

Parameters

- **from_fmk** (*bool*) – set to True if the call was performed by the framework itself, otherwise the call comes from user-code (e.g., from a *probe* or an *operator*)
- **data** (Data) – data container that embeds generally a modeled data accessible through *data.content*. However if the latter is None, it only embeds the raw data.

send_data_sync(*data:* framework.data.Data, *from_fmk=False*)
Can be used in user-code to send data to the target without interfering with the framework.

Use case example: The user needs to send some message to the target on a regular basis in background. For that purpose, it can quickly define a *framework.monitor.Probe* that just emits the message by itself.

send_multiple_data(*data_list*, *from_fmk=False*)
Used to send multiple data to the target, or to stimulate several target's inputs in one shot.

Note: Use data.to_bytes() to get binary data

Parameters

- **from_fmk** (*bool*) – set to True if the call was performed by the framework itself, otherwise the call comes from user-code (e.g., from a *Probe* or an *Operator*)
- **data_list** (*list*) – list of data to be sent

send_multiple_data_sync(*data_list*, *from_fmk=False*)
Can be used in user-code to send data to the target without interfering with the framework.

send_pending_data(*from_fmk=False*)

sending_delay = 0

set_data_model(*dm*)

set_feedback_mode(*mode*)

set_feedback_timeout(*fbk_timeout*)
To set dynamically the feedback timeout.

Parameters **fbk_timeout** (*float*) – maximum time duration for collecting the feedback

set_logger(*logger*)

set_project(*prj*)

set_sending_delay(*sending_delay*)
Set the sending delay.

Parameters **sending_delay** (*float*) – maximum time (in seconds) taken to send data once the method `send_(multiple_)data()` has been called.

start()
To be overloaded if needed

stop()
To be overloaded if needed

supported_feedback_mode = [1, 2]

exception framework.target_helpers.TargetError
Bases: Exception

__module__ = 'framework.target_helpers'

exception framework.target_helpers.TargetNotReady
Bases: Exception

__module__ = 'framework.target_helpers'

exception framework.target_helpers.TargetStuck
Bases: Exception

__module__ = 'framework.target_helpers'

13.2.9 framework.targets.network module

class framework.targets.network.NetworkTarget(*host='localhost'*, *port=12345*, *socket_type=(<AddressFamily.AF_INET: 2>, <SocketKind.SOCK_STREAM: 1>)*, *data_semantics='Unknown Semantic'*, *server_mode=False*, *listen_on_start=True*, *target_address=None*, *wait_for_client=True*, *hold_connection=False*, *keep_first_client=True*, *mac_src=None*, *mac_dst=None*, *add_eth_header=False*, *fbk_timeout=2*, *fbk_mode=1*, *sending_delay=1*, *recover_timeout=0.5*)
Bases: *framework.target_helpers.Target*

Generic target class for interacting with a network resource. Can be used directly, but some methods may require to be overloaded to fit your needs.

CHUNK_SZ = 2048

General_Info_ID = 'General Information'

UNKNOWN_SEMANTIC = 'Unknown Semantic'

_INTERNALS_ID = 'NetworkTarget()'

__annotations__ = {}

__init__(*host='localhost'*, *port=12345*, *socket_type=(<AddressFamily.AF_INET: 2>, <SocketKind.SOCK_STREAM: 1>)*, *data_semantics='Unknown Semantic'*, *server_mode=False*, *listen_on_start=True*, *target_address=None*, *wait_for_client=True*, *hold_connection=False*, *keep_first_client=True*, *mac_src=None*, *mac_dst=None*, *add_eth_header=False*, *fbk_timeout=2*, *fbk_mode=1*, *sending_delay=1*, *recover_timeout=0.5*)

Parameters

- **host** (*str*) – IP address of the target to connect to, or the IP address on which we will wait for target connecting to us (if *server_mode* is True). For raw socket type, it should contain the name of the interface.

- **port** (*int*) – Port for communicating with the target, or the port to listen to. For raw socket type, it should contain the protocol ID.
- **socket_type** (*tuple*) – Tuple composed of the socket address family and socket type
- **data_semantics** (*str*) – String of characters that will be used for data routing decision. Useful only when more than one interface are defined. In such case, the data semantics will be checked in order to find a matching interface to which data will be sent. If the data have no semantic, it will be routed to the default first declared interface.
- **server_mode** (*bool*) – If *True*, the interface will be set in server mode, which means we will wait for the real target to connect to us for sending it data.
- **listen_on_start** (*bool*) – If *True*, servers will be launched right after the *NetworkTarget* starts. Otherwise, they will be launched in a lazy mode, meaning just when something is about to be sent through the server mode interface.
- **target_address** (*tuple*) – Used only if *server_mode* is *True* and socket type is *SOCK_DGRAM*. To be used if data has to be sent to a specific address (which is not necessarily the client). It is especially useful if you need to send data before receiving anything. What should be provided is a tuple *(host(str), port(int))* associated to the target.
- **wait_for_client** (*bool*) – Used only in server mode (*server_mode* is *True*) when the *socket type* is *SOCK_DGRAM* and a *target_address* is provided, or when the *socket_type* is *SOCK_RAW*. If set to *True*, before sending any data, the *NetworkTarget* will wait for the reception of data (from any client); otherwise it will send data as soon as provided.
- **hold_connection** (*bool*) – If *True*, we will maintain the connection while sending data to the real target. Otherwise, after each data emission, we close the related socket.
- **keep_first_client** (*bool*) – Used only in server mode (*server_mode* is *True*) with *SOCK_STREAM* socket type. If set to *True*, the first client that connects to the server will remain the one used for data sending until the target is reloaded. Otherwise, last client information are used. This is not supported for *SOCK_DGRAM* where the first client will always be the one used for data sending.
- **mac_src** (*bytes*) – Only in conjunction with raw socket. For each data sent through this interface, and if this data contain nodes with the semantic `'mac_src'`, these nodes will be overwritten (through absorption) with this parameter. If nothing is provided, the MAC address will be retrieved from the interface specified in 'host'. (works accurately for Linux system).
- **mac_dst** (*bytes*) – Only in conjunction with raw socket. For each data sent through this interface, and if this data contain nodes with the semantic `'mac_dst'`, these nodes will be overwritten (through absorption) with this parameter.
- **add_eth_header** (*bool*) – Add an ethernet header to the data to send. Only possible in combination with a SOCK_RAW socket type.
- **fbk_timeout** (*float*) – maximum time duration for collecting the feedback
- **sending_delay** (*float*) – maximum time (in seconds) taken to send data once the method `send_(multiple_)data()` has been called.
- **recover_timeout** (*int*) – Allowed delay for recovering the target. (the recovering can be triggered by the framework if the feedback threads did not terminate before the target health check) Impact the behavior of self.recover_target().

__module__ = 'framework.targets.network'

_before_sending_data(*data_list*, *from_fmk*)

_cleanup_state()

_collect_feedback_from(*thread_id*, *fbk_sockets*, *fbk_ids*, *fbk_lengths*, *epobj*, *fileno2fd*, *fbk_timeout*, *flush_received_fbk*, *pre_fbk*)

_connect_to_additional_feedback_sockets()
Connection to additional feedback sockets, if any.

_connect_to_target(*host*, *port*, *socket_type*)

_custom_data_handling_before_emission(*data_list*)
To be overloaded if you want to perform some operation before sending *data_list* to the target.

Parameters **data_list** (*list*) – list of Data objects that will be sent to the target.

Returns the data list to send

Return type list

_feedback_collect(*fbk*, *ref*, *error=0*)

_feedback_complete()

_feedback_handling(*fbk*, *ref*)
To be overloaded if feedback from the target need to be filtered before being logged and/or collected in some way and/or for any other reasons.

Parameters

- **fbk** (*bytes*) – feedback received by the target through a socket referenced by *ref*.
- **ref** (*string*) – user-defined reference of the socket used to retrieve the feedback.

Returns

a tuple *(new_fbk, status)* where *new_fbk* is the feedback you want to log and *status* is a status that enables you to notify a problem to the framework (should be positive if everything is fine, otherwise should be negative).

Return type tuple

_feedback_mode = 1

_get_additional_feedback_sockets()
Used if any additional socket to get feedback from has been added by *NetworkTarget.add_additional_feedback_interface()*, related to the data emitted if needed.

Parameters **data** (Data) – the data that will be sent.

Returns

list of sockets, dict of associated ids/names, dict of associated length (a length can be None)

Return type tuple

_get_data_semantic_key(*data*)

_get_net_info_from(*data*)

_get_socket_type(*host*, *port*)

_handle_connection_to_fbk_server(*clientsocket*, *address*, *args*, *pre_fbk=None*)

_handle_target_connection(*clientsocket*, *address*, *args*, *pre_fbk=None*)

_is_valid_socket_type(*socket_type*)

_listen_to_target(*host*, *port*, *socket_type*, *func*, *args=None*)

_raw_connect_to(*host*, *port*, *ref_id*, *socket_type=(<AddressFamily.AF_INET: 2>, <SocketKind.SOCK_STREAM: 1>)*, *chk_size=2048*, *hold_connection=True*)

_raw_listen_to(*host*, *port*, *ref_id*, *socket_type=(<AddressFamily.AF_INET: 2>, <SocketKind.SOCK_STREAM: 1>)*, *chk_size=2048*, *wait_time=None*)

_raw_server_main(*serversocket*, *host*, *port*, *sock_type*, *func*, *sending_event*, *notif_host_event*)

_register_last_ack_date(*ack_date*)

_send_data(*sockets*, *data_refs*, *fbk_timeout*, *from_fmk*, *pre_fbk=None*)

_server_main(*serversocket*, *host*, *port*, *func*)

_start_fbk_collector(*fbk_sockets*, *fbk_ids*, *fbk_lengths*, *epobj*, *fileno2fd*, *pre_fbk=None*, *timeout=None*, *flush_received_fbk=False*)

add_additional_feedback_interface(*host*, *port*, *socket_type=(<AddressFamily.AF_INET: 2>, <SocketKind.SOCK_STREAM: 1>)*, *fbk_id=None*, *fbk_length=None*, *server_mode=False*, *wait_time=None*)

Allows to register additional socket to get feedback from. Connection is attempted be when target starts, that is when *NetworkTarget.start()* is called.

cleanup()

To be overloaded if something needs to be performed after each data emission. It is called after any feedback has been retrieved.

collect_unsolicited_feedback(*timeout=0*)

If overloaded, it should collect any data from the associated real target that may be sent without solicitation (i.e. without any data sent through it) and make it available through the method .get_feedback()

Parameters **timeout** – Maximum delay before returning from feedback collecting

Returns False if it is not possible, otherwise it should be True

Return type bool

connect_to(*host*, *port*, *ref_id*, *socket_type=(<AddressFamily.AF_INET: 2>, <SocketKind.SOCK_STREAM: 1>)*, *chk_size=2048*, *hold_connection=True*)

Used for collecting feedback from the target while it is already started.

get_description()

get_feedback()

If overloaded, should return a FeedbackCollector object.

get_last_target_ack_date()

If different from None the return value is used by the FMK to log the date of the target acknowledgment after a message has been sent to it.

[Note: If this method is overloaded, is_feedback_received() should also be]

initialize()

To be overloaded if some intial setup for the target is necessary.

is_feedback_received()

To be overloaded if the target implements FBK_WAIT_UNTIL_RECV mode, so that it can informs the framework about feedback reception.

is_target_ready_for_new_data()

To be overloaded if the target needs some time (for conditions to occur) before data can be sent. Note: The FMK busy wait on this method() before sending a new data.

listen_to(*host*, *port*, *ref_id*, *socket_type=(<AddressFamily.AF_INET: 2>, <SocketKind.SOCK_STREAM: 1>)*, *chk_size=2048*, *wait_time=None*, *hold_connection=True*)
Used for collecting feedback from the target while it is already started.

recover_target()
Implementation of target recovering operations, when a target problem has been detected (i.e. a negative feedback from a probe, an operator or the Target() itself)

Returns True if the target has been recovered. False otherwise.

Return type bool

register_new_interface(*host*, *port*, *socket_type*, *data_semantics*, *server_mode=False*, *target_address=None*, *wait_for_client=True*, *hold_connection=False*, *keep_first_client=True*, *mac_src=None*, *mac_dst=None*, *add_eth_header=False*)

remove_all_dynamic_interfaces()

remove_dynamic_interface(*host*, *port*)

send_data(*data*, *from_fmk=False*)
To be overloaded.

Note: use data.to_bytes() to get binary data.

Parameters

- **from_fmk** (*bool*) – set to True if the call was performed by the framework itself, otherwise the call comes from user-code (e.g., from a *probe* or an *operator*)
- **data** (Data) – data container that embeds generally a modeled data accessible through *data.content*. However if the latter is None, it only embeds the raw data.

send_multiple_data(*data_list*, *from_fmk=False*)
Used to send multiple data to the target, or to stimulate several target's inputs in one shot.

Note: Use data.to_bytes() to get binary data

Parameters

- **from_fmk** (*bool*) – set to True if the call was performed by the framework itself, otherwise the call comes from user-code (e.g., from a *Probe* or an *Operator*)
- **data_list** (*list*) – list of data to be sent

set_timeout(*fbk_timeout*, *sending_delay*)
Set the time duration for feedback gathering and the sending delay above which we give up: - sending data to the target (client mode) - waiting for client connections before sending data to them (server mode)

Parameters

- **fbk_timeout** – time duration for feedback gathering (in seconds)
- **sending_delay** – sending delay (in seconds)

start()
To be overloaded if needed

stop()
To be overloaded if needed

supported_feedback_mode = [1, 2]

terminate()
To be overloaded if some cleanup is necessary for stopping the target.

13.2.10 framework.targets.local module

class framework.targets.local.**LocalTarget**(*target_path=None*, *pre_args=None*, *post_args=None*, *tmpfile_ext='.bin'*, *send_via_stdin=False*, *send_via_cmdline=False*, *error_samples=None*, *error_parsing_func=<function LocalTarget.<lambda>>*)

Bases: *framework.target_helpers.Target*

__annotations__ = {}

__init__(*target_path=None*, *pre_args=None*, *post_args=None*, *tmpfile_ext='.bin'*, *send_via_stdin=False*, *send_via_cmdline=False*, *error_samples=None*, *error_parsing_func=<function LocalTarget.<lambda>>*)

__module__ = 'framework.targets.local'

_before_sending_data()

_feedback_mode = 2

cleanup()
To be overloaded if something needs to be performed after each data emission. It is called after any feedback has been retrieved.

get_description()

get_feedback(*timeout=0.2*)
If overloaded, should return a FeedbackCollector object.

get_post_args()

get_pre_args()

get_target_path()

initialize()
To be overloaded if some intial setup for the target is necessary.

send_data(*data*, *from_fmk=False*)
To be overloaded.

Note: use data.to_bytes() to get binary data.

Parameters

- **from_fmk** (*bool*) – set to True if the call was performed by the framework itself, otherwise the call comes from user-code (e.g., from a *probe* or an *operator*)
- **data** (*Data*) – data container that embeds generally a modeled data accessible through *data.content*. However if the latter is None, it only embeds the raw data.

set_post_args(*post_args*)

set_pre_args(*pre_args*)

set_target_path(*target_path*)

set_tmp_file_extension(*tmpfile_ext*)

start()
To be overloaded if needed

stop()
To be overloaded if needed

supported_feedback_mode = [2]

terminate()
To be overloaded if some cleanup is necessary for stopping the target.

13.2.11 framework.targets.sim module

13.2.12 framework.targets.ssh module

class framework.targets.ssh.SSHTarget(*target_addr='localhost'*, *port=12345*, *bind_address=None*, *username=None*, *password=None*, *pkey_path=None*, *pkey_password=None*, *proxy_jump_addr=None*, *proxy_jump_bind_addr=None*, *proxy_jump_port=None*, *proxy_jump_username=None*, *proxy_jump_password=None*, *proxy_jump_pkey_path=None*, *proxy_jump_pkey_password=None*, *targeted_command=None*, *file_parameter_path=None*, *fbk_timeout=0.5*, *read_stdout=True*, *read_stderr=True*, *char_mapping=False*, *get_pty=False*, *ref=None*)

Bases: *framework.target_helpers.Target*

ASK_PASSWORD = 20

NO_PASSWORD = 10

STATUS_THRESHOLD_FOR_RECOVERY = -2

__annotations__ = {}

__init__(*target_addr='localhost'*, *port=12345*, *bind_address=None*, *username=None*, *password=None*, *pkey_path=None*, *pkey_password=None*, *proxy_jump_addr=None*, *proxy_jump_bind_addr=None*, *proxy_jump_port=None*, *proxy_jump_username=None*, *proxy_jump_password=None*, *proxy_jump_pkey_path=None*, *proxy_jump_pkey_password=None*, *targeted_command=None*, *file_parameter_path=None*, *fbk_timeout=0.5*, *read_stdout=True*, *read_stderr=True*, *char_mapping=False*, *get_pty=False*, *ref=None*)

This generic target enables you to interact with a remote target requiring an SSH connection.

Parameters

- **target_addr** – IP address to reach the SSH server
- **port** – port on which the SSH server listen to.
- **bind_address** – source address for communication.
- **username** – username to use for the connection.
- **password** – (optional) password related to the username. Could also be the special value *SSHTarget.ASK_PASSWORD* that will prompt the user for the password at the time of connection.
- **pkey_path** – (optional) path to the private key related to the username (if no password provided).
- **pkey_password** – (optional) if the private key is encrypted, this parameter can be either the password to decrypt it, or the special value *SSHTarget.ASK_PASSWORD* that will prompt the user for the password at the time of connection. If the private key is not encrypted, then this parameter should be set to *SSHTarget.NO_PASSWORD*
- **proxy_jump_addr** – If a proxy jump has to be done before reaching the target, this parameter should be provided with the proxy address to connect with.
- **proxy_jump_bind_addr** – internal address of the proxy to communication with the target.

- **proxy_jump_port** – port on which the SSH server of the proxy listen to.
- **proxy_jump_username** – username to use for the connection with the proxy.
- **proxy_jump_password** – (optional) password related to the username. Could also be the special value *SSHTarget.ASK_PASSWORD* that will prompt the user for the password at the time of connection.
- **proxy_jump_pkey_path** – (optional) path to the private key related to the username.
- **proxy_jump_pkey_password** – (optional) if the private key is encrypted, this parameter can be either the password to decrypt it, or the special value *SSHTarget.ASK_PASSWORD* that will prompt the user for the password at the time of connection. If the private key is not encrypted, then this parameter should be set to *SSHTarget.NO_PASSWORD*.
- **targeted_command** – If not None, it should be a format string taking one argument that will be automatically filled either with the data to be sent or with @file_parameter_path if it is not None (meaning the data have to be provided through a file).
- **file_parameter_path** – If data should be provided to the targeted command through a file, then this parameter should provide the remote path where the data to be sent will be first copied into (otherwise it should remain equal to None). it will be provided as a parameter of @targeted_command.
- **fbk_timeout** – delay for the framework to wait before it requests feedback from us.
- **read_stdout** (*bool*) – If *True*, collect as feedback what the executed command will write in stdout.
- **read_stderr** (*bool*) – If *True*, collect as feedback what the executed command will write in stderr.
- **char_mapping** (*dict*) – If provided, specific characters in the payload will be replaced based on it.
- **get_pty** (*bool*) – Request a pseudo-terminal from the server.
- **ref** (*str*) – Reference for the target. Used for description only.

__module__ = 'framework.targets.ssh'

_set_feedback_timeout_specific(*fbk_timeout*)

Overload this function to handle feedback specifics

Parameters **fbk_timeout** (*float*) – time duration for collecting the feedback

get_description()

get_last_target_ack_date()

If different from None the return value is used by the FMK to log the date of the target acknowledgment after a message has been sent to it.

[Note: If this method is overloaded, is_feedback_received() should also be]

is_feedback_received()

To be overloaded if the target implements FBK_WAIT_UNTIL_RECV mode, so that it can informs the framework about feedback reception.

recover_target()

Implementation of target recovering operations, when a target problem has been detected (i.e. a negative feedback from a probe, an operator or the Target() itself)

Returns True if the target has been recovered. False otherwise.

Return type bool

send_data(*data*, *from_fmk=False*)
To be overloaded.

Note: use data.to_bytes() to get binary data.

Parameters

- **from_fmk** (*bool*) – set to True if the call was performed by the framework itself, otherwise the call comes from user-code (e.g., from a *probe* or an *operator*)
- **data** (Data) – data container that embeds generally a modeled data accessible through *data.content*. However if the latter is None, it only embeds the raw data.

start()
To be overloaded if needed

stop()
To be overloaded if needed

13.2.13 framework.targets.printer module

class framework.targets.printer.**PrinterTarget**(*tmpfile_ext*)
Bases: *framework.target_helpers.Target*

__annotations__ = {}

__init__(*tmpfile_ext*)

__module__ = 'framework.targets.printer'

_feedback_mode = None

get_description()

get_printer_name()

get_target_ip()

get_target_port()

send_data(*data*, *from_fmk=False*)
To be overloaded.

Note: use data.to_bytes() to get binary data.

Parameters

- **from_fmk** (*bool*) – set to True if the call was performed by the framework itself, otherwise the call comes from user-code (e.g., from a *probe* or an *operator*)
- **data** (Data) – data container that embeds generally a modeled data accessible through *data.content*. However if the latter is None, it only embeds the raw data.

set_printer_name(*printer_name*)

set_target_ip(*target_ip*)

set_target_port(*target_port*)

set_tmp_file_extension(*tmpfile_ext*)

start()
To be overloaded if needed

supported_feedback_mode = []

13.2.14 framework.targets.debug module

exception framework.targets.debug.**IncorrectTargetError**
Bases: Exception

__module__ = 'framework.targets.debug'

exception framework.targets.debug.**ShmemMappingError**
Bases: Exception

__module__ = 'framework.targets.debug'

class framework.targets.debug.**TestTarget**(*name=None*, *recover_ratio=100*, *fbk_samples=None*, *repeat_input=False*, *fbk_timeout=0.05*, *shmem_mode=False*, *shmem_timeout=10*)
Bases: *framework.target_helpers.Target*

__annotations__ = {}

__init__(*name=None*, *recover_ratio=100*, *fbk_samples=None*, *repeat_input=False*, *fbk_timeout=0.05*, *shmem_mode=False*, *shmem_timeout=10*)

__module__ = 'framework.targets.debug'

_collect_fbk_loop()

_feedback_mode = 1

_forward_data()

_handle_fbk(*data*)

_last_ack_date = None

_map_input_shmem()

add_binding(*target*)

add_feedback_sources(**targets*)

consumer_start = 6

consumer_stop = 15

data_start = 16

dlen_format = '>L'

dlen_start = 0

dlen_stop = 4

get_consumer_idx()

get_feedback()
If overloaded, should return a FeedbackCollector object.

get_last_target_ack_date()
If different from None the return value is used by the FMK to log the date of the target acknowledgment after a message has been sent to it.

[Note: If this method is overloaded, is_feedback_received() should also be]

is_feedback_received()
To be overloaded if the target implements FBK_WAIT_UNTIL_RECV mode, so that it can informs the framework about feedback reception.

is_target_ready_for_new_data()
To be overloaded if the target needs some time (for conditions to occur) before data can be sent. Note: The FMK busy wait on this method() before sending a new data.

max_consumer = 10

meta_data_size = 16

producer_status_idx = 5

recover_target()
Implementation of target recovering operations, when a target problem has been detected (i.e. a negative feedback from a probe, an operator or the Target() itself)

Returns True if the target has been recovered. False otherwise.

Return type bool

send_data(*data*, *from_fmk=False*)
To be overloaded.

Note: use data.to_bytes() to get binary data.

Parameters

- **from_fmk** (*bool*) – set to True if the call was performed by the framework itself, otherwise the call comes from user-code (e.g., from a *probe* or an *operator*)
- **data** (Data) – data container that embeds generally a modeled data accessible through *data.content*. However if the latter is None, it only embeds the raw data.

send_multiple_data(*data_list*, *from_fmk=False*)
Used to send multiple data to the target, or to stimulate several target's inputs in one shot.

Note: Use data.to_bytes() to get binary data

Parameters

- **from_fmk** (*bool*) – set to True if the call was performed by the framework itself, otherwise the call comes from user-code (e.g., from a *Probe* or an *Operator*)
- **data_list** (*list*) – list of data to be sent

set_control_delay(*delay*)

set_control_over(**test_targets*, *feedback_filter=<function TestTarget.<lambda>>*)

shmem_size = 4096

start()
To be overloaded if needed

stop()
To be overloaded if needed

supported_feedback_mode = [2, 1]

13.2.15 framework.project module

class framework.project.**Project**(*enable_fbk_processing=True*, *wkspace_enabled=True*, *wkspace_size=1000*, *wkspace_free_slot_ratio_when_full=0.5*, *fmkdb_enabled=True*, *default_fbk_timeout=None*, *default_fbk_mode=None*, *default_sending_delay=None*, *default_burst_value=None*)

Bases: object

__init__(*enable_fbk_processing=True*, *wkspace_enabled=True*, *wkspace_size=1000*, *wkspace_free_slot_ratio_when_full=0.5*, *fmkdb_enabled=True*, *default_fbk_timeout=None*, *default_fbk_mode=None*, *default_sending_delay=None*, *default_burst_value=None*)

Parameters

- **enable_fbk_processing** – enable or disable the execution of feedback handlers, if any are set in the project.
- **wkspace_enabled** – If set to True, enable the framework workspace that store the generated data.
- **wkspace_size** – Maximum number of data that can be stored in the workspace.
- **wkspace_free_slot_ratio_when_full** – when the workspace is full, provide the ratio of the workspace size that will be used as the amount of entries to free in the workspace.
- **fmkdb_enabled** – If set to *True*, the fmkDB will be used. Otherwise, no DB transactions will occur and thus the fmkDB won't be filled during the session.
- **default_fbk_timeout** – If not None, when the project will be run, this value will be used to initialize the feedback timeout of all the targets
- **default_fbk_mode** – If not None, when the project will be run, this value will be used to initialize the feedback mode of all the targets
- **default_sending_delay** – If not None, when the project will be run, this value will be used to initialize the delay that is applied by the framework between each data sending.
- **default_burst_value** – If not None, when the project will be run, this value will be used to initialize the burst value of the framework (number of data that can be sent in burst before a delay is applied).

__module__ = 'framework.project'

_feedback_processing()
core function of the feedback processing thread

add_knowledge(**info*)

default_dm = None

disable_feedback_handlers()

enable_feedback_handlers()

estimate_last_data_impact_uniqueness()

get_operator(*name*)

get_operators()

get_probes()

property knowledge_source

map_targets_to_scenario(*scenario*, *target_mapping*)

name = None

notify_data_sending(*data_list*, *timestamp*, *target*)

register_evolutionary_processes(**processes*)

register_feedback_handler(*fbk_handler*)

register_operator(*name*, *obj*)

register_probe(*probe*, *blocking=False*)

register_scenarios(**scenarios*)

reset_knowledge()

reset_target_mappings()

set_data_model(*dm*)

set_exportable_fmk_ops(*fmkops*)

set_logger(*logger*)

set_monitor(*monitor*)

set_targets(*targets*)

share_knowlegde_source()

start()

stop()

trigger_feedback_handlers(*source*, *timestamp*, *content*, *status*)

wkspace_enabled = None

wkspace_free_slot_ratio_when_full = None

wkspace_size = None

13.2.16 framework.operator_helpers module

class framework.operator_helpers.**LastInstruction**

Bases: object

RecordData = 1

__init__()

__module__ = 'framework.operator_helpers'

get_comments()

get_operator_feedback()

get_operator_status()

get_timestamp()

is_instruction_set(*name*)

set_comments(*comments*)

set_instruction(*name*)

set_operator_feedback(*info*)

set_operator_status(*status_code*)

class framework.operator_helpers.**Operation**

Bases: object

CleanupDMakers = 3

Exportable = 2

Stop = 1

__init__()

__module__ = 'framework.operator_helpers'

add_instruction(*actions*, *seed=None*, *tg_ids=None*)

get_instructions()

is_flag_set(*name*)

set_flag(*name*)

set_status(*status*)

class framework.operator_helpers.**Operator**

Bases: object

__module__ = 'framework.operator_helpers'

__str__()

Return str(self).

_args_desc = None

_start(*fmk_ops*, *dm*, *monitor*, *target*, *logger*, *user_input*)

do_after_all(*fmk_ops*, *dm*, *monitor*, *target*, *logger*)

This action is executed after data has been sent to the target AND that all blocking probes have returned. BUT just before data is logged.

Returns

Last-minute instructions you request fuddly to perform.

Return type *LastInstruction*

plan_next_operation(*fmk_ops*, *dm*, *monitor*, *target*, *logger*, *fmk_feedback*)

Shall return a Operation object that contains the operations that you want fuddly to perform.

Returns Operation you want fuddly to perform.

Return type *Operation*

start(*fmk_ops*, *dm*, *monitor*, *target*, *logger*, *user_input*)

To be overloaded if specific initialization code is needed. Shall return True if setup has succeeded, otherwise shall return False.

stop(*fmk_ops*, *dm*, *monitor*, *target*, *logger*)

To be overloaded if specific termination code is needed.

framework.operator_helpers.**operator**(*prj*, *args=None*)

13.2.17 framework.logger module

class framework.logger.**Logger**(*name=None*, *prefix=''*, *record_data=False*, *explicit_data_recording=False*, *export_raw_data=True*, *term_display_limit=800*, *enable_term_display=True*, *enable_file_logging=False*, *highlight_marked_nodes=False*)

Bases: object

The Logger is used for keeping the history of the communication with the Target. The methods are used by the framework, but can also be leveraged by an Operator.

FLUSH_API = 1

PRETTY_PRINT_API = 3

PRINT_CONSOLE_API = 4

WRITE_API = 2

__init__(*name=None*, *prefix=''*, *record_data=False*, *explicit_data_recording=False*, *export_raw_data=True*, *term_display_limit=800*, *enable_term_display=True*, *enable_file_logging=False*, *highlight_marked_nodes=False*)

Parameters

- **name** (*str*) – Name to be used in the log filenames. If not specified, the name of the project in which the logger is embedded will be used.
- **record_data** (*bool*) – If True, each emitted data will be stored in a specific file within *exported_data/*.
- **explicit_data_recording** (*bool*) – Used for logging outcomes further to an Operator instruction. If True, the operator would have to state explicitly if it wants the just emitted data to be recorded. Such notification is possible when the framework call its method *framework.operator_helpers.Operator.do_after_all()*, where the Operator can take its decision after the observation of the target feedback and/or probes outputs.
- **export_raw_data** (*bool*) – If True, will log the data as it is, without trying to interpret it as human readable text.
- **term_display_limit** (*int*) – maximum amount of characters to display on the terminal at once. If this threshold is overrun, the message to print on the console will be truncated.
- **enable_term_display** (*bool*) – If True, information will be displayed on the terminal
- **prefix** (*str*) – prefix to use for printing on the console.
- **enable_file_logging** (*bool*) – If True, file logging will be enabled.
- **highlight_marked_nodes** (*bool*) – If True, alteration performed by compatible disruptors will be highlighted. Only possible if *export_raw_data* is False, as this option forces data interpretation.

__module__ = 'framework.logger'

__str__()
Return str(self).

_encode_target_feedback(*feedback*)

_export_data_func(*data*, *suffix=''*)

_handle_binary_content(*content*, *raw=False*)

_log_feedback(*source*, *content*, *status_code*, *timestamp*, *record=True*)

_log_handler()

_print_console(*msg*, *nl_before=True*, *nl_after=False*, *rgb=None*, *style=None*, *raw_limit=None*, *limit_output=True*, *no_format_mode=False*)

_process_target_feedback(*feedback*)

_stop_log_handler()

collect_feedback(*content*, *status_code=None*, *subref=None*, *fbk_src=None*)

Used within the scope of the Logger feedback-collector infrastructure. If your target implement the interface `Target.get_feedback()`, no need to use this infrastructure.

To be called by the target each time feedback need to be registered.

Parameters

- **content** – feedback record
- **status_code** (*int*) – should be negative for error
- **subref** (*str*) – specific reference to distinguish internal log sources within the same caller
- **fbk_src** – [optional] source object of the feedback

commit_data_table_entry(*group_id*, *prj_name*)

flush()

fmkDB = None

log_async_data(*data_list: Union[*framework.data.Data*, List[*framework.data.Data*], Tuple[*framework.data.Data*]]*, *sent_date*, *target_ref*, *prj_name*, *current_data_id*)

log_collected_feedback(*preamble=None*, *epilogue=None*)

Used within the scope of the Logger feedback-collector feature. If your target implement the interface `Target.get_feedback()`, no need to use this infrastructure.

It allows to retrieve the collected feedback, that has been populated by the target (through call to *Logger.collect_feedback()*).

Parameters

- **preamble** (*str*) – prefix added to each collected feedback
- **epilogue** (*str*) – suffix added to each collected feedback

Returns

True if target feedback has been collected through logger infrastructure *Logger.collect_feedback()*, False otherwise.

Return type bool

log_comment(*comment*)

log_data(*data*, *verbose=False*)

log_data_info(*data_info*, *dmaker_type*, *data_maker_name*)

log_disruptor_info(*dmaker_type*, *name*, *user_input*)

log_dmaker_step(*num*)

log_error(*err_msg*)

log_fmk_info(*info*, *nl_before=False*, *nl_after=False*, *rgb=6750207*, *data_id=None*, *do_show=True*, *do_record=True*, *delay_recording=False*)

log_generator_info(*dmaker_type*, *name*, *user_input*, *data_id=None*, *disabled=False*)

log_info(*info*)

log_operator_feedback(*operator*, *content*, *status_code*, *timestamp*)

log_probe_feedback(*probe*, *content*, *status_code*, *timestamp*, *related_tg=None*)

log_target_ack_date()

log_target_feedback_from(*source*, *content*, *status_code*, *timestamp*, *preamble=None*, *epilogue=None*)

pretty_print_data(*data:* framework.data.Data, *fd=None*, *raw_limit: Optional[int] = None*)

print_console(*msg*, *nl_before=True*, *nl_after=False*, *rgb=None*, *style=None*, *raw_limit=None*, *limit_output=True*, *no_format_mode=False*)

reset_current_state()

set_external_display(*disp*)

set_target_ack_date(*tg_ref*, *date*)

shall_record()

start()

start_new_log_entry(*preamble=''*)

stop()

wait_for_sync()

write(*data: str*)

13.2.18 framework.monitor module

exception framework.monitor.**AddExistingProbeToMonitorError**(*probe_name*)

Bases: Exception

Raised when a probe is being added a second time in a monitor

__init__(*probe_name*)

__module__ = 'framework.monitor'

property probe_name

class framework.monitor.**BlockingProbeUser**(*probe*, *after_target_feedback_retrieval*)

Bases: *framework.monitor.ProbeUser*

__init__(*probe*, *after_target_feedback_retrieval*)

__module__ = 'framework.monitor'

_clear()

Clear all events

_notify_armed()

_notify_status_retrieved()

_run(**args*, ***kwargs*)

_wait_for_data_ready()

Wait on a request to arm

Returns

True if the arm event happened, False if a stop was asked or an error was signaled

Return type bool

_wait_for_fmk_sync()
Wait on a blocking event: data sent or timeout

Returns

True if the blocking event happened, False if a stop was asked or an error was signaled

Return type bool

property after_target_feedback_retrieval

notify_blocking()

notify_data_ready()

notify_error()
Informs the probe of an error

stop()

wait_until_armed(*timeout=None*)

wait_until_ready(*timeout=None*)

class framework.monitor.**Monitor**
Bases: object

__init__()

__module__ = 'framework.monitor'

_get_probe_ref(*probe*)

_wait_for_specific_probes(*probe_user_class*, *probe_user_wait_method*, *probes=None*, *timeout=None*)
Wait for probes to trigger a specific event

Parameters

- **probe_user_class** (ProbeUser) – probe_user class that defines the method.
- **probe_user_wait_method** (*method*) – name of the probe_user's method that will be used to wait.
- **probes** (list of *ProbeUser*) – probes to wait for. If None all probes will be concerned.
- **timeout** (*float*) – maximum time to wait for in seconds.

add_probe(*probe*, *blocking=False*, *after_target_feedback_retrieval=False*)

configure_probe(*probe*, **args*)

disable_hooks()

enable_hooks()

get_probe_delay(*probe*)

get_probe_related_tg(*probe*)

get_probe_status(*probe*)

get_probes_names()

is_probe_launched(*probe*)

is_probe_stuck(*probe*)

is_target_ok()

iter_probes()

notify_data_sending_event()

notify_error()

notify_imminent_data_sending()

notify_target_feedback_retrieval()

set_data_model(*dm*)

set_fmk_ops(*fmk_ops*)

set_logger(*logger*)

set_probe_delay(*probe*, *delay*)

set_targets(*targets*)

start()

start_probe(*probe*, *related_tg=None*)

stop()

stop_all_probes()

stop_probe(*probe*)

property target_status

wait_for_probe_initialization()

wait_for_probe_status_retrieval()

class framework.monitor.**Probe**(*delay=1.0*)

Bases: object

__init__(*delay=1.0*)

__module__ = 'framework.monitor'

__str__()

Return str(self).

_start(*dm*, *target*, *logger*)

_stop(*dm*, *target*, *logger*)

arm(*dm*, *target*, *logger*)

Only used by blocking probes. Called by the framework just before sending a data.

Parameters

- **dm** – the current data model
- **target** – the current target
- **logger** – the current logger

configure(**args*)

(Optional method) To be overloaded with any signature that fits your needs Could be called by user code through *framework.monitor.Monitor.configure_probe()* Use case example is to call it from an *framework.operator_helpers.Operator*

Parameters ***args** – anything that fits your needs

property delay

main(*dm*, *target*, *logger*)
To be overloaded by user-code

In the case of a basic probe, this method will be called in loop following a period specified within the associated project file.

In the case of a blocking probe, this method will be called by the framework just after having sent a data (or a batch of data).

Parameters

- **dm** – the current data model
- **target** – the current target
- **logger** – the current logger

Returns negative status if something is wrong

Return type *ProbeStatus*

reset()
To be overloaded by user-code (if needed).

Called each time the probe status is retrieved by the framework (through *Monitor.get_probe_status()*). Useful especially for periodic probe that may need to be reset after each data sending.

Note: shall be stateless and reentrant.

start(*dm*, *target*, *logger*)
Probe initialization

Returns may return a status or None

Return type *ProbeStatus*

property status

stop(*dm*, *target*, *logger*)

class framework.monitor.**ProbeCmd**
Bases: *framework.monitor.Probe*

Generic probe that enables you to execute shell commands and retrieve the output.

The monitoring can be done through different backend (e.g., SSH_Backend, Serial_Backend).

backend
backend to be used (e.g., SSH_Backend).

Type *framework.comm_backends.Backend*

init_command
ssh command to execute at init

Type str

recurrent_command
ssh command to execute at each probing

Type str

__annotations__ = {}

__init__()

__module__ = 'framework.monitor'

backend = None

init_command = None

main(*dm*, *target*, *logger*)
To be overloaded by user-code

In the case of a basic probe, this method will be called in loop following a period specified within the associated project file.

In the case of a blocking probe, this method will be called by the framework just after having sent a data (or a batch of data).

Parameters

- **dm** – the current data model
- **target** – the current target
- **logger** – the current logger

Returns negative status if something is wrong

Return type *ProbeStatus*

recurrent_command = None

start(*dm*, *target*, *logger*)
Probe initialization

Returns may return a status or None

Return type *ProbeStatus*

stop(*dm*, *target*, *logger*)

class framework.monitor.**ProbeMem**
Bases: *framework.monitor.Probe*

Generic probe that enables you to monitor the process memory (RSS...) consumption. It can be done by specifying a `threshold` and/or a `tolerance` ratio.

The monitoring can be done through different backend (e.g., `SSH_Backend`, `Serial_Backend`).

backend
backend to be used (e.g., `SSH_Backend`).

Type *framework.comm_backends.Backend*

process_name
name of the process to monitor.

Type str

threshold
memory (RSS) threshold that the monitored process should not exceed. (dimension should be the same as what is provided by the *ps* command of the system under test)

Type int

tolerance
tolerance expressed in percentage of the memory (RSS) the process was using at the beginning of the monitoring (or after each time the tolerance has been exceeded).

Type int

command_pattern
format string for the ssh command. '{0:s}' refer to the process name.

Type str

__annotations__ = {}

__init__()

__module__ = 'framework.monitor'

_get_mem()

backend = None

command_pattern = 'ps -e -orss,comm | grep {0:s}'

main(*dm*, *target*, *logger*)
To be overloaded by user-code

In the case of a basic probe, this method will be called in loop following a period specified within the associated project file.

In the case of a blocking probe, this method will be called by the framework just after having sent a data (or a batch of data).

Parameters

- **dm** – the current data model
- **target** – the current target
- **logger** – the current logger

Returns negative status if something is wrong

Return type *ProbeStatus*

process_name = None

reset()
To be overloaded by user-code (if needed).

Called each time the probe status is retrieved by the framework (through *Monitor.get_probe_status()*). Useful especially for periodic probe that may need to be reset after each data sending.

Note: shall be stateless and reentrant.

start(*dm*, *target*, *logger*)
Probe initialization

Returns may return a status or None

Return type *ProbeStatus*

stop(*dm*, *target*, *logger*)

threshold = None

tolerance = 2

class framework.monitor.**ProbePID**
Bases: *framework.monitor.Probe*

Generic probe that enables you to monitor a process PID.

The monitoring can be done through different backend (e.g., SSH_Backend, Serial_Backend).

backend
backend to be used (e.g., SSH_Backend).

Type *framework.comm_backends.Backend*

process_name
name of the process to monitor.

Type str

max_attempts
maximum number of attempts for getting the process ID.

Type int

delay_between_attempts
delay in seconds between each attempt.

Type float

delay
delay before retrieving the process PID.

Type float

command_pattern
format string for the ssh command. '{0:s}' refer to the process name.

Type str

__annotations__ = {}

__init__()

__module__ = 'framework.monitor'

_get_pid(*logger*)

backend = None

command_pattern = 'pgrep {0:s}'

delay = 0.5

delay_between_attempts = 0.1

main(*dm*, *target*, *logger*)
To be overloaded by user-code

In the case of a basic probe, this method will be called in loop following a period specified within the associated project file.

In the case of a blocking probe, this method will be called by the framework just after having sent a data (or a batch of data).

Parameters

- **dm** – the current data model
- **target** – the current target
- **logger** – the current logger

Returns negative status if something is wrong

Return type *ProbeStatus*

max_attempts = 10

process_name = None

start(*dm*, *target*, *logger*)
Probe initialization

Returns may return a status or None

Return type *ProbeStatus*

stop(*dm*, *target*, *logger*)

class framework.monitor.**ProbeStatus**(*status=None*, *info=None*)
Bases: object

__init__(*status=None*, *info=None*)

__module__ = 'framework.monitor'

get_private_info()

get_timestamp()

set_private_info(*pv*)

set_timestamp()

property value

exception framework.monitor.**ProbeTimeoutError**(*probe_name*, *timeout*, *blocking_methods=None*)
Bases: Exception

Raised when a probe is considered stuck

__init__(*probe_name*, *timeout*, *blocking_methods=None*)

Parameters

- **probe_name** (*str*) – name of the probe where the timeout occurred
- **timeout** (*float*) – time the probe waited before its timeout
- **blocking_methods** (*list of str*) – list of probe_methods where the timeout may have happened

__module__ = 'framework.monitor'

property blocking_methods

property probe_name

property timeout

class framework.monitor.**ProbeUser**(*probe*)
Bases: object

__annotations__ = {}

__init__(*probe*)

__module__ = 'framework.monitor'

_clear()
Clear all events

_go_on()

_handle_exception(*context*)

_notify_probe_started()

_run(**args*, ***kwargs*)

_wait(*delay*)

_wait_for_probe(*event*, *timeout=None*)
Wait for the probe to trigger a specific event

get_probe_delay()

get_probe_status()

is_alive()

is_stuck()
Tells if the probe has to be considered stuck by the monitor: i.e. if it is really stuck or if its stop was not acknowledged

join(*timeout=None*)

property probe

probe_init_timeout = 15.0

set_probe_delay(*delay*)

start(**args*, ***kwargs*)

stop()

timeout = 5.0

wait_for_probe_init(*timeout=None*)

framework.monitor.**blocking_probe**(*project*, *after_target_feedback_retrieval=False*)

framework.monitor.**probe**(*project*)

13.2.19 framework.comm_backends module

class framework.comm_backends.**Backend**(*codec='latin_1'*)
Bases: object

__init__(*codec='latin_1'*)

Parameters **codec** (*str*) – codec used by the monitored system to answer.

__module__ = 'framework.comm_backends'

_exec_command(*cmd*)

Parameters **cmd** – command to execute through the communication channel

Returns: list of file descriptors (e.g., stdout, stderr)

_start()

_stop()

exec_command(*cmd*)

read_output(*chan_desc*)

Parameters **chan_desc** – object returned by *Backend.exec_command()* that enables to gather output data

Returns data retrieved through the communication channel

Return type bytes

start()

stop()

exception framework.comm_backends.**BackendError**(*msg*, *status=- 1*)

Bases: Exception

__init__(*msg*, *status=- 1*)

__module__ = 'framework.comm_backends'

class framework.comm_backends.**SSH_Backend**(*target_addr='localhost'*, *port=22*, *bind_address=None*, *username=None*, *password=None*, *pkey_path=None*, *pkey_password=10*, *proxy_jump_addr=None*, *proxy_jump_bind_addr=None*, *proxy_jump_port=None*, *proxy_jump_username=None*, *proxy_jump_password=None*, *proxy_jump_pkey_path=None*, *proxy_jump_pkey_password=10*, *codec='latin-1'*, *timeout=None*, *get_pty=False*)

Bases: *framework.comm_backends.Backend*

ASK_PASSWORD = 20

Backend to execute command through a serial line.

NO_PASSWORD = 10

__annotations__ = {}

__init__(*target_addr='localhost'*, *port=22*, *bind_address=None*, *username=None*, *password=None*, *pkey_path=None*, *pkey_password=10*, *proxy_jump_addr=None*, *proxy_jump_bind_addr=None*, *proxy_jump_port=None*, *proxy_jump_username=None*, *proxy_jump_password=None*, *proxy_jump_pkey_path=None*, *proxy_jump_pkey_password=10*, *codec='latin-1'*, *timeout=None*, *get_pty=False*)

Parameters

- **target_addr** (*str*) – IP of the SSH server.
- **port** (*int*) – port of the SSH server.
- **username** (*str*) – username to connect with.
- **password** (*str*) – (optional) password related to the username. Could also be the special value *SSHTarget.ASK_PASSWORD* that will prompt the user for the password at the time of connection.
- **pkey_path** (*str*) – (optional) path of the private key (if no password provided).
- **pkey_password** – (optional) if the private key is encrypted, this parameter can be either the password to decrypt it, or the special value *SSHTarget.ASK_PASSWORD* that will prompt the user for the password at the time of connection. If the private key is not encrypted, then this parameter should be set to *SSHTarget.NO_PASSWORD*

- **proxy_jump_addr** – If a proxy jump has to be done before reaching the target, this parameter should be provided with the proxy address to connect with.
- **proxy_jump_bind_addr** – internal address of the proxy to communication with the target.
- **proxy_jump_port** – port on which the SSH server of the proxy listen to.
- **proxy_jump_username** – username to use for the connection with the proxy.
- **proxy_jump_password** – (optional) password related to the username. Could also be the special value *SSHTarget.ASK_PASSWORD* that will prompt the user for the password at the time of connection.
- **proxy_jump_pkey_path** – (optional) path to the private key related to the username.
- **proxy_jump_pkey_password** – (optional) if the private key is encrypted, this parameter can be either the password to decrypt it, or the special value *SSHTarget.ASK_PASSWORD* that will prompt the user for the password at the time of connection. If the private key is not encrypted, then this parameter should be set to *SSHTarget.NO_PASSWORD*.
- **codec** (*str*) – codec used by the monitored system to answer.
- **timeout** (*float*) – timeout on blocking read/write operations. None disables timeouts on socket operations
- **get_pty** (*bool*) – Request a pseudo-terminal from the server. It implies that processes executed from this ssh session will be attached to the pty and will be killed once the session is closed. (Otherwise they could remain on the server.)

__module__ = 'framework.comm_backends'

static _create_pkey(*pkey_path*, *pkey_password*, *prompt='PKey Password:'*)

_exec_command(*cmd*)

Parameters **cmd** – command to execute through the communication channel

Returns: list of file descriptors (e.g., stdout, stderr)

_read_fd(*fdesc*)

_start()

_stop()

read_output(*chan_desc*)

Parameters **chan_desc** – object returned by *Backend.exec_command()* that enables to gather output data

Returns data retrieved through the communication channel

Return type bytes

read_stderr(*chan_desc*)

read_stdout(*chan_desc*)

set_timeout(*timeout*)

class framework.comm_backends.**Serial_Backend**(*serial_port*, *baudrate=115200*, *bytesize=8*, *parity='N'*, *stopbits=1*, *xonxoff=False*, *rtscts=False*, *dsrdtr=False*, *username=None*, *password=None*, *slowness_factor=5*, *cmd_notfound=b'command not found'*, *codec='latin-1'*)

Bases: *framework.comm_backends.Backend*

Backend to execute command through a serial line.

__annotations__ = {}

__init__(*serial_port*, *baudrate=115200*, *bytesize=8*, *parity='N'*, *stopbits=1*, *xonxoff=False*, *rtscts=False*, *dsrdtr=False*, *username=None*, *password=None*, *slowness_factor=5*, *cmd_notfound=b'command not found'*, *codec='latin-1'*)

Parameters

- **serial_port** (*str*) – path to the tty device file. (e.g., '/dev/ttyUSB0')
- **baudrate** (*int*) – baud rate of the serial line.
- **bytesize** (*int*) – number of data bits. (5, 6, 7, or 8)
- **parity** (*str*) – parity checking. ('N', 'O, 'E', 'M', or 'S')
- **stopbits** (*int*) – number of stop bits. (1, 1.5 or 2)
- **xonxoff** (*bool*) – enable software flow control.
- **rtscts** (*bool*) – enable hardware (RTS/CTS) flow control.
- **dsrdtr** (*bool*) – enable hardware (DSR/DTR) flow control.
- **username** (*str*) – username to connect with. If None, no authentication step will be attempted.
- **password** (*str*) – password related to the username.
- **slowness_factor** (*int*) – characterize the slowness of the monitored system. The scale goes from 1 (fastest) to 10 (slowest). This factor is a base metric to compute the time to wait for the authentication step to terminate (if *username* and *password* parameter are provided) and other operations involving to wait for the monitored system.
- **cmd_notfound** (*bytes*) – pattern used to detect if the command does not exist on the monitored system.
- **codec** (*str*) – codec used to send/receive information through the serial line

__module__ = 'framework.comm_backends'

_exec_command(*cmd*)

Parameters **cmd** – command to execute through the communication channel

Returns: list of file descriptors (e.g., stdout, stderr)

_read_serial(*serial_chan*, *duration*)

_start()

_stop()

read_output(*chan_desc*)

Parameters **chan_desc** – object returned by *Backend.exec_command()* that enables to gather output data

Returns data retrieved through the communication channel

Return type bytes

class framework.comm_backends.**Shell_Backend**(*timeout=None*, *codec='latin-1'*)

Bases: *framework.comm_backends.Backend*

Backend to execute shell commands locally

__annotations__ = {}

__init__(*timeout=None*, *codec='latin-1'*)

Parameters

- **timeout** (*float*) – timeout in seconds for reading the result of the command
- **codec** (*str*) – codec used by the monitored system to answer.

__module__ = 'framework.comm_backends'

_exec_command(*cmd*)

Parameters **cmd** – command to execute through the communication channel

Returns: list of file descriptors (e.g., stdout, stderr)

_start()

_stop()

read_output(*chan_desc*)

Parameters **chan_desc** – object returned by *Backend.exec_command()* that enables to gather output data

Returns data retrieved through the communication channel

Return type bytes

13.2.20 framework.tactics_helpers module

class framework.tactics_helpers.**DataMaker**

Bases: object

__annotations__ = {}

__init__()

__module__ = 'framework.tactics_helpers'

_args_desc = None

_modelwalker_user = False

knowledge_source = None

property modelwalker_user

related_dm_name = None

set_exportable_fmk_ops(*fmkops*)

class framework.tactics_helpers.DataMakerAttr
Bases: object

Active = 1

Controller = 2

HandOver = 3

NeedSeed = 5

SetupRequired = 4

__module__ = 'framework.tactics_helpers'

class framework.tactics_helpers.Disruptor
Bases: *framework.tactics_helpers.DataMaker*

__annotations__ = {}

__init__()

__module__ = 'framework.tactics_helpers'

_cleanup()

_setup(*dm*, *user_input*)

cleanup(*fmkops*)
–> Specific code

clear_attr(*name*)

disrupt_data(*dm*, *target*, *prev_data*)

is_attr_set(*name*)

set_attr(*name*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.tactics_helpers.DynGenerator
Bases: *framework.tactics_helpers.Generator*

__annotations__ = {}

__module__ = 'framework.tactics_helpers'

_args_desc = {'determinist': ("Make the data model determinist if set to 'True', random if set to 'False', or do nothing if set to 'None'", None, <class 'bool'>), 'finite': ('Make the data model finite', False, <class 'bool'>), 'freeze': ('Freeze the generated node.', False, <class 'bool'>), 'leaf_determinism': ("If set to 'True', all the typed nodes of the model will be set to determinist mode prior to any fuzzing. If set to 'False', they will be set to random mode. Otherwise, if set to 'None', nothing will be done.", None, <class 'bool'>), 'min_def': ("Set the default quantity of all the nodes to the defined minimum quantity if this parameter is set to 'True', or maximum quantity if set to 'False'. Otherwise if set to 'None', nothing is done.", None, <class 'bool'>), 'resolve_csp': ('Resolve any CSP if any', True, <class 'bool'>)}

data_id = ''

generate_data(*dm*, *monitor*, *target*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.tactics_helpers.**DynGeneratorFromScenario**
Bases: *framework.tactics_helpers.Generator*

__annotations__ = {}

__handle_transition_callbacks(*hook*, *feedback=None*)

__module__ = 'framework.tactics_helpers'

_alter_data_step()

_alter_transition_conditions()

_args_desc = {'cond_fuzz': ('For each scenario step having guarded transitions, a new scenario is created where transition conditions are inverted. [compatible with ignore_timing]', False, <class 'bool'>), 'data_fuzz': ('For each scenario step that generates data, a new scenario is created where the data generated by the step is fuzzed.', False, <class 'bool'>), 'graph': ('Display the scenario and highlight the current step each time the generator is called.', False, <class 'bool'>), 'graph_format': ('Format to be used for displaying the scenario (e.g., xdot, pdf, png).', 'xdot', <class 'str'>), 'ignore_timing': ('For each scenario step enforcing a timing constraint, a new scenario is created where any timeout conditions are removed (i.e., set to 0 second). [compatible with cond_fuzz]', False, <class 'bool'>), 'init': ("Used in combination with 'data_fuzz', 'cond_fuzz', or 'ignore_timing'. Make the generator begin with the Nth corrupted scenario (where N is provided through this parameter).", 0, <class 'int'>), 'reset': ("If set, scenarios created by 'data_fuzz', 'cond_fuzz', or 'ignore_timing' will reinitialize the scenario after each corruption case, without waiting for the normal continuation of the scenario.", True, <class 'bool'>), 'stutter': ("For each scenario step that generates data, a new scenario is created where the step is altered to stutter 'stutter_max' times, meaning that data-sending steps would be triggered 'stutter_max' times.", False, <class 'bool'>), 'stutter_max': ("The number of times a step will stutter [to be used with 'stutter']", 2, <class 'int'>)}

_callback_dispatcher_after_fbk(*fbk*)
This callback is always called by the framework It allows for a NoDataStep to perform actions (trigger periodic data, tasks, ...)

_callback_dispatcher_after_sending()

_callback_dispatcher_before_sending_step1()

_callback_dispatcher_before_sending_step2()

_callback_dispatcher_final()

_check_data_fuzz_completion_cbk(*env*, *step*)

_cleanup_walking_attrs()

_make_step_stutter()

_stutter_cbk(*env*, *current_step*, *next_step*)

cleanup(*fmkops*)
–> Specific code

generate_data(*dm*, *monitor*, *target*)

graph_scenario(*fmt*, *select_current=False*)

property produced_seed

scenario = None

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.tactics_helpers.Generator
Bases: framework.tactics_helpers.DataMaker

__annotations__ = {}

__init__()

__module__ = 'framework.tactics_helpers'

_cleanup()

_setup(*dm*, *user_input*)

cleanup(*fmkops*)
–> Specific code

clear_attr(*name*)

generate_data(*dm*, *monitor*, *target*)

is_attr_set(*name*)

need_reset()

produced_seed = None

set_attr(*name*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.tactics_helpers.StatefulDisruptor
Bases: framework.tactics_helpers.DataMaker

__annotations__ = {}

__init__()

__module__ = 'framework.tactics_helpers'

_cleanup()

_set_seed(*prev_data*)

_setup(*dm*, *user_input*)

cleanup(*fmkops*)
–> Specific code

clear_attr(*name*)

disrupt_data(*dm*, *target*, *data*)
@data: it is either equal to prev_data the first time disrupt_data() is called by the FMK, or it is a an empty data (that is Data()).

handover()

is_attr_set(*name*)

set_attr(*name*)

set_seed(*prev_data*)

setup(*dm*, *user_input*)
–> Specific code return True if setup has succeeded, otherwise return False

class framework.tactics_helpers.**Tactics**
Bases: object

__clear_dmaker_clones(*dmaker*, *dmaker_clones*)

__clone_dmaker(*dmaker*, *dmaker_clones*, *dmaker_type*, *new_dmaker_type*, *dmaker_name=None*, *register_func=None*)

__get_random_data_maker(*dict_var*, *dmaker_type*, *total_weight*, *valid*)

__init__()

__module__ = 'framework.tactics_helpers'

__register_new_data_maker(*dict_var*, *name*, *obj*, *weight*, *dmaker_type*, *valid*)

__set_data_maker_weight(*dict_var*, *dmaker_type*, *name*, *weight*)

clear_disruptor_clones()

clear_generator_clones()

clone_disruptor(*dmaker_type*, *new_dmaker_type=None*, *dmaker_name=None*)

clone_generator(*dmaker_type*, *new_dmaker_type=None*, *dmaker_name=None*)

property disruptor_types

disruptors_info()

property generator_types

generators_info()

get_datatype_total_weight(*dmaker_type*)

get_disruptor_name(*dmaker_type*, *obj*)

get_disruptor_obj(*dmaker_type*, *name*)

get_disruptor_validness(*dmaker_type*, *name*)

get_disruptor_weight(*dmaker_type*, *name*)

get_disruptors_list(*dmaker_type*)

get_dmaker_type_total_weight(*dmaker_type*)

get_generator_name(*dmaker_type*, *obj*)

get_generator_obj(*dmaker_type*, *name*)

get_generator_validness(*dmaker_type*, *name*)

get_generator_weight(*dmaker_type*, *name*)

get_generators_list(*dmaker_type*)

get_info_from_obj(*obj*)

get_random_disruptor(*dmaker_type*, *valid*)

get_random_generator(*dmaker_type*, *valid*)

print_disruptor(*dmaker_type*, *disruptor_name*)

print_generator(*dmaker_type*, *generator_name*)

register_new_disruptor(*name*, *obj*, *weight*, *dmaker_type*, *valid=False*)

register_new_generator(*name*, *obj*, *weight*, *dmaker_type*, *valid=False*)

register_scenarios(**scenarios*)

static scenario_ref_from(*scenario*)

set_additional_info(*fmkops*, *related_dm=None*)

set_disruptor_weight(*dmaker_type*, *name*, *weight*)

set_generator_weight(*dmaker_type*, *name*, *weight*)

framework.tactics_helpers.**_handle_user_inputs**(*dmaker*, *user_input*)

framework.tactics_helpers.**_restore_dmaker_internals**(*dmaker*)

framework.tactics_helpers.**_user_input_conformity**(*self*, *user_input*, *_args_desc*)

framework.tactics_helpers.**disruptor**(*st*, *dtype*, *weight=1*, *valid=False*, *args=None*, *modelwalker_user=False*)

class framework.tactics_helpers.**dyn_generator**(*name*, *bases*, *attrs*)

Bases: type

__annotations__ = {}

__init__(*name*, *bases*, *attrs*)

__module__ = 'framework.tactics_helpers'

data_id = ''

class framework.tactics_helpers.**dyn_generator_from_scenario**(*name*, *bases*, *attrs*)

Bases: type

__annotations__ = {}

__module__ = 'framework.tactics_helpers'

static __new__(*cls*, *name*, *bases*, *attrs*)

scenario = None

framework.tactics_helpers.**generator**(*st*, *gtype*, *weight=1*, *valid=False*, *args=None*, *modelwalker_user=False*)

framework.tactics_helpers.**modelwalker_inputs_handling_helper**(*dmaker*)

13.2.21 framework.fuzzing_primitives module

class framework.fuzzing_primitives.**AltConfConsumer**(*max_runs_per_node=- 1*, *min_runs_per_node=-1*, *respect_order=True*, *fuzz_magnitude=1.0*, *fix_constraints=False*, *ignore_mutable_attr=False*, *consider_side_effects_on_sibbling=False*, ***kwargs*)

Bases: *framework.fuzzing_primitives.NodeConsumerStub*

Note: save_node()/restore_node() are not overloaded although default implementation can triggers overhead, because for some cases copying the Elt is the better (e.g., for alternate conf on nonterm nodes, that reuse same subnodes over the various confs).

__module__ = 'framework.fuzzing_primitives'

consume_node(*node*)
Use this method to modify/alter or just read information on @node. This function will be called for each node that satisfy the criteria. (to be implemented according to the implementation of need_reset())

Return True to say that you have correctly consumed the node. Return False, if despite your current criteria for node interest, you are in fact not interested

init_specific(***kwargs*)

need_reset(*node*)

recover_node(*node*)
Generic way to recover a node

save_node(*node*)
Generic way to save a node (can impact performance)

still_interested_by(*node*)

wait_for_exhaustion(*node*)

- return -1 to wait until exhaustion
- return 0 to stop node iteration after consumption (and yielding a value once)
- return N-1 to stop iteration after at most N step (or before if exhaustion triggers)

class framework.fuzzing_primitives.**BasicVisitor**(*max_runs_per_node=- 1*, *min_runs_per_node=- 1*, *respect_order=True*, *fuzz_magnitude=1.0*, *fix_constraints=False*, *ignore_mutable_attr=False*, *consider_side_effects_on_sibbling=False*, ***kwargs*)
Bases: *framework.fuzzing_primitives.NodeConsumerStub*

__annotations__ = {}

__module__ = 'framework.fuzzing_primitives'

consume_node(*node*)
Use this method to modify/alter or just read information on @node. This function will be called for each node that satisfy the criteria. (to be implemented according to the implementation of need_reset())

Return True to say that you have correctly consumed the node. Return False, if despite your current criteria for node interest, you are in fact not interested

init_specific(*reset_when_change=True*)

need_reset(*node*)

recover_node(*node*)
Generic way to recover a node

reset_state()
Called by the ModelWalker to reinitialize the disruptor.

save_node(*node*)
Generic way to save a node (can impact performance)

wait_for_exhaustion(*node*)

- return -1 to wait until exhaustion
- return 0 to stop node iteration after consumption (and yielding a value once)
- return N-1 to stop iteration after at most N step (or before if exhaustion triggers)

class framework.fuzzing_primitives.**ModelWalker**(*root_node*, *node_consumer*, *make_determinist=False*, *make_random=False*, *max_steps=- 1*, *initial_step=1*)

Bases: object

We walk through all states of the model and give opportunity to the Consumer to act on each node, and to be involved in the walking process in some extents.

The first rule of the walking process is to step up to a node exhaustion (which means that the consume_node() method of the Consumer won't be called in-between)

Note: the change of a non-terminal node does not reset the indirect parents (just the direct parent), otherwise it could lead to a combinatorial explosion, with limited interest...

__init__(*root_node*, *node_consumer*, *make_determinist=False*, *make_random=False*, *max_steps=- 1*, *initial_step=1*)

__iter__()

__module__ = 'framework.fuzzing_primitives'

_do_reset(*node*, *consumer*)

node_consumer_helper(*node*, *structure_has_changed*, *consumed_nodes*, *parent_node*, *consumer*)

set_consumer(*node_consumer*)

walk_graph_rec(*node_list*, *structure_has_changed*, *consumed_nodes*, *parent_node*, *consumer*)

class framework.fuzzing_primitives.**NodeConsumerStub**(*max_runs_per_node=- 1*, *min_runs_per_node=- 1*, *respect_order=True*, *fuzz_magnitude=1.0*, *fix_constraints=False*, *ignore_mutable_attr=False*, *consider_side_effects_on_sibbling=False*, ***kwargs*)

Bases: object

__annotations__ = {}

__init__(*max_runs_per_node=- 1*, *min_runs_per_node=- 1*, *respect_order=True*, *fuzz_magnitude=1.0*, *fix_constraints=False*, *ignore_mutable_attr=False*, *consider_side_effects_on_sibbling=False*, ***kwargs*)

__module__ = 'framework.fuzzing_primitives'

consume_node(*node*)

Use this method to modify/alter or just read information on @node. This function will be called for each node that satisfy the criteria. (to be implemented according to the implementation of need_reset())

Return True to say that you have correctly consumed the node. Return False, if despite your current criteria for node interest, you are in fact not interested

do_after_reset(*node*)

init_specific(***kwargs*)

interested_by(*node*)

max_nb_runs_for(*node*)

need_reset(*node*)

preload(*root_node*)

Called by the ModelWalker when it initializes

Parameters **root_node** – Root node of the modeled data

Returns: None

recover_node(*node*)
Generic way to recover a node

reset_state()
Called by the ModelWalker to reinitialize the disruptor.

save_node(*node*)
Generic way to save a node (can impact performance)

set_node_interest(*internals_criteria=None*, *semantics_criteria=None*, *owned_confs=None*, *path_regexp=None*, *conf=None*)
@conf: criteria are applied for the provided conf if not None, otherwise current_conf is used Note: when all is None, NodeConsumer is interested by every node (that is interested_by() return always True)

still_interested_by(*node*)

wait_for_exhaustion(*node*)

- return -1 to wait until exhaustion
- return 0 to stop node iteration after consumption (and yielding a value once)
- return N-1 to stop iteration after at most N step (or before if exhaustion triggers)

class framework.fuzzing_primitives.**NonTermVisitor**(*max_runs_per_node=- 1*, *min_runs_per_node=- 1*, *respect_order=True*, *fuzz_magnitude=1.0*, *fix_constraints=False*, *ignore_mutable_attr=False*, *consider_side_effects_on_sibbling=False*, ***kwargs*)

Bases: *framework.fuzzing_primitives.BasicVisitor*

__annotations__ = {}

__module__ = 'framework.fuzzing_primitives'

consume_node(*node*)
Use this method to modify/alter or just read information on @node. This function will be called for each node that satisfy the criteria. (to be implemented according to the implementation of need_reset())

Return True to say that you have correctly consumed the node. Return False, if despite your current criteria for node interest, you are in fact not interested

init_specific(*reset_when_change=True*)

need_reset(*node*)

still_interested_by(*node*)

wait_for_exhaustion(*node*)

- return -1 to wait until exhaustion
- return 0 to stop node iteration after consumption (and yielding a value once)
- return N-1 to stop iteration after at most N step (or before if exhaustion triggers)

class framework.fuzzing_primitives.**SeparatorDisruption**(*max_runs_per_node=- 1*, *min_runs_per_node=- 1*, *respect_order=True*, *fuzz_magnitude=1.0*, *fix_constraints=False*, *ignore_mutable_attr=False*, *consider_side_effects_on_sibbling=False*, ***kwargs*)

Bases: *framework.fuzzing_primitives.NodeConsumerStub*

__annotations__ = {}

__module__ = 'framework.fuzzing_primitives'

consume_node(*node*)

Use this method to modify/alter or just read information on @node. This function will be called for each node that satisfy the criteria. (to be implemented according to the implementation of need_reset())

Return True to say that you have correctly consumed the node. Return False, if despite your current criteria for node interest, you are in fact not interested

init_specific(*separators=None*)

class framework.fuzzing_primitives.**TypedNodeDisruption**(*max_runs_per_node=- 1*, *min_runs_per_node=- 1*, *respect_order=True*, *fuzz_magnitude=1.0*, *fix_constraints=False*, *ignore_mutable_attr=False*, *consider_side_effects_on_sibbling=False*, ***kwargs*)

Bases: *framework.fuzzing_primitives.NodeConsumerStub*

__annotations__ = {}

__module__ = 'framework.fuzzing_primitives'

_add_separator_cases(*vt_node*)

_populate_fuzzy_vt_list(*vt_node*, *fuzz_magnitude*)

consume_node(*node*)

Use this method to modify/alter or just read information on @node. This function will be called for each node that satisfy the criteria. (to be implemented according to the implementation of need_reset())

Return True to say that you have correctly consumed the node. Return False, if despite your current criteria for node interest, you are in fact not interested

init_specific(*ignore_separator=False*, *determinist=True*)

need_reset(*node*)

preload(*root_node*)

Called by the ModelWalker when it initializes

Parameters **root_node** – Root node of the modeled data

Returns: None

recover_node(*node*)

Generic way to recover a node

save_node(*node*)

Generic way to save a node (can impact performance)

still_interested_by(*node*)

framework.fuzzing_primitives.**fuzz_data_tree**(*top_node*, *paths_regexp=None*)

13.2.22 framework.encoders module

class framework.encoders.**BitReverse_Enc**(*encoding_arg=None*)

Bases: *framework.encoders.Encoder*

__module__ = 'framework.encoders'

_reverse_bits(*x*, *nb_bits=8*)

Reverse bits order of x

decode(*val*)

To be overloaded. (Should be stateless.)

Parameters **val** (*bytes*) – the encoded value

Returns the decoded value

Return type bytes

encode(*val*)

To be overloaded. (Should be stateless.)

Parameters **val** (*bytes*) – the value

Returns the encoded value

Return type bytes

class framework.encoders.**Encoder**(*encoding_arg=None*)

Bases: object

__annotations__ = {}

__copy__()

__init__(*encoding_arg=None*)

__module__ = 'framework.encoders'

decode(*val*)

To be overloaded. (Should be stateless.)

Parameters **val** (*bytes*) – the encoded value

Returns the decoded value

Return type bytes

encode(*val*)

To be overloaded. (Should be stateless.)

Parameters **val** (*bytes*) – the value

Returns the encoded value

Return type bytes

init_encoding_scheme(*arg*)

To be optionally overloaded by a subclass that deals with encoding, if encoding need to be initialized in some way. (called at init and in String.reset())

Parameters **arg** – provided through the *encoding_arg* parameter of the *String* constructor

reset()

class framework.encoders.**GSM7bitPacking_Enc**(*encoding_arg=None*)
Bases: *framework.encoders.Encoder*

__annotations__ = {}

__module__ = 'framework.encoders'

decode(*msg*)
To be overloaded. (Should be stateless.)

Parameters **val** (*bytes*) – the encoded value

Returns the decoded value

Return type bytes

encode(*msg*)
To be overloaded. (Should be stateless.)

Parameters **val** (*bytes*) – the value

Returns the encoded value

Return type bytes

class framework.encoders.**GSMPhoneNum_Enc**(*encoding_arg=None*)
Bases: *framework.encoders.Encoder*

__annotations__ = {}

__module__ = 'framework.encoders'

decode(*msg*)
To be overloaded. (Should be stateless.)

Parameters **val** (*bytes*) – the encoded value

Returns the decoded value

Return type bytes

encode(*msg*)
To be overloaded. (Should be stateless.)

Parameters **val** (*bytes*) – the value

Returns the encoded value

Return type bytes

class framework.encoders.**GZIP_Enc**(*encoding_arg=None*)
Bases: *framework.encoders.Encoder*

__annotations__ = {}

__module__ = 'framework.encoders'

decode(*val*)
To be overloaded. (Should be stateless.)

Parameters **val** (*bytes*) – the encoded value

Returns the decoded value

Return type bytes

encode(*val*)
To be overloaded. (Should be stateless.)

Parameters **val** (*bytes*) – the value

Returns the encoded value

Return type bytes

init_encoding_scheme(*arg=None*)
To be optionally overloaded by a subclass that deals with encoding, if encoding need to be initialized in some way. (called at init and in `String.reset()`)

Parameters **arg** – provided through the *encoding_arg* parameter of the *String* constructor

class framework.encoders.**Wrap_Enc**(*encoding_arg=None*)
Bases: *framework.encoders.Encoder*

Encoder to be used as a mean to wrap a Node with a prefix and/or a suffix, without defining specific Nodes for that (meaning you don't need to model that part and want to simplify your data description).

__annotations__ = {}

__module__ = 'framework.encoders'

decode(*val*)
To be overloaded. (Should be stateless.)

Parameters **val** (*bytes*) – the encoded value

Returns the decoded value

Return type bytes

encode(*val*)
To be overloaded. (Should be stateless.)

Parameters **val** (*bytes*) – the value

Returns the encoded value

Return type bytes

init_encoding_scheme(*arg*)
Take a list parameter specifying the prefix and the suffix to add to the value to encode, or to remove from an encoded value.

Parameters **arg** (*list*) – Prefix and suffix character strings. Can be individually set to None

13.2.23 framework.database module

class framework.database.**Database**(*fmkdb_path=None*)
Bases: `object`

DDL_fname = 'fmk_db.sql'

DEFAULT_DB_NAME = 'fmkDB.db'

DEFAULT_DM_NAME = '__DEFAULT_DATAMODEL'

DEFAULT_GEN_NAME = '__DEFAULT_GNAME'

DEFAULT_GTYPE_NAME = '__DEFAULT_GTYPE'

FEEDBACK_TRAIL_TIME_WINDOW = 10

OUTCOME_DATA = 2

OUTCOME_ROWID = 1

__init__(*fmkdb_path=None*)

__module__ = 'framework.database'

_get_color_function(*colorized*)

_handle_binary_content(*content*, *sz_limit=None*, *raw=False*, *colorized=True*)

_is_valid(*connection*, *cursor*)

_sql_handler()

_stop_sql_handler()

check_data_existence(*data_id*, *colorized=True*)

column_names_from(*table*)

disable()

display_data_info(*data_id*, *with_data=False*, *with_fbk=False*, *with_fmkinfo=True*, *with_analysis=True*, *with_async_data=False*, *fbk_src=None*, *limit_data_sz=None*, *page_width=100*, *colorized=True*, *raw=False*, *decoding_hints=None*, *dm_list=None*)

display_data_info_by_date(*start*, *end*, *with_data=False*, *with_fbk=False*, *with_fmkinfo=True*, *with_analysis=True*, *with_async_data=False*, *fbk_src=None*, *prj_name=None*, *limit_data_sz=None*, *raw=False*, *page_width=100*, *colorized=True*, *decoding_hints=None*, *dm_list=None*)

display_data_info_by_range(*first_id*, *last_id*, *with_data=False*, *with_fbk=False*, *with_fmkinfo=True*, *with_analysis=True*, *with_async_data=False*, *fbk_src=None*, *prj_name=None*, *limit_data_sz=None*, *raw=False*, *page_width=100*, *colorized=True*, *decoding_hints=None*, *dm_list=None*)

display_stats(*colorized=True*)

enable()

execute_sql_statement(*sql_stmt*, *params=None*)

export_data(*first*, *last=None*, *colorized=True*)

fetch_data(*start_id=1*, *end_id=- 1*)

flush_current_feedback()

flush_feedback()

get_data_with_impact(*prj_name=None*, *fbk_src=None*, *fbk_status_formula='? < 0'*, *display=True*, *verbose=False*, *raw_analysis=False*, *colorized=True*)

get_data_with_specific_fbk(*fbk*, *prj_name=None*, *fbk_src=None*, *display=True*, *colorized=True*)

get_data_without_fbk(*prj_name=None*, *fbk_src=None*, *display=True*, *colorized=True*)

static get_default_db_path()

get_next_data_id(*prev_id=None*)

get_project_record(*prj_name=None*)

insert_analysis(*data_id*, *content*, *date*, *impact=False*)

insert_async_data(*dtype*, *dm_name*, *raw_data*, *sz*, *sent_date*, *target_ref*, *prj_name*, *current_data_id=None*)

insert_comment(*data_id*, *content*, *date*)

insert_data(*dtype*, *dm_name*, *raw_data*, *sz*, *sent_date*, *ack_date*, *target_ref*, *prj_name*, *group_id=None*)

insert_data_model(*dm_name*)

insert_dmaker(*dm_name*, *dtype*, *name*, *is_gen*, *stateful*, *clone_type=None*)

insert_feedback(*data_id*, *source*, *timestamp*, *content*, *status_code=None*)

insert_fmk_info(*data_id*, *content*, *date*, *error=False*)

insert_project(*prj_name*)

insert_steps(*data_id*, *step_id*, *dmaker_type*, *dmaker_name*, *data_id_src*, *user_input*, *info*)

is_enabled()

iter_feedback_entries(*last=True*, *source=None*)

remove_data(*data_id*, *colorized=True*)

shrink_db()

start()

stop()

submit_sql_stmt(*stmt*, *params=None*, *outcome_type: Optional[int] = None*, *error_msg=''*)

This method is the only one that should submit request to the threaded SQL handler. It is also synchronized to guarantee request order (especially needed when you wait for the outcomes of your submitted SQL statement).

Parameters

- **stmt** (*str*) – SQL statement
- **params** (*tuple*) – parameters
- **outcome_type** (*int*) – type of the expected outcomes. If *None*, no outcomes are expected
- **error_msg** (*str*) – specific error message to display in case of an error

Returns *None* or the expected outcomes

class framework.database.**FeedbackGate**(*database*, *only_last_entries=True*)

Bases: object

__bool__()

__init__(*database*, *only_last_entries=True*)

Parameters **database** (Database) – database to be associated with

__iter__()

__module__ = 'framework.database'

get_feedback_from(*source*)

iter_entries(*source=None*)

Iterate over feedback entries that are related to the last data which has been sent by the framework.

Parameters **source** (FeedbackSource) – feedback source to consider

Returns

A generator that iterates over all the requested feedback entries and provides for each:

- the triplet: (status, timestamp, content) if *source* is associated to a specific feedback source

- the 4-uplet: (source, status, timestamp, content) if *source* is *None*

Return type python generator

property size

sources_names()

Return a list of the feedback source names related to the last data which has been sent by the framework.

Returns names of the feedback sources

Return type list

framework.database.**regexp**(*expr*, *item*)

framework.database.**regexp_bin**(*expr*, *item*)

13.2.24 framework.scenario module

class framework.scenario.**FinalStep**(*data_desc=None*, *final=False*, *fbk_timeout=None*, *fbk_mode=None*, *sending_delay=None*, *set_periodic=None*, *clear_periodic=None*, *step_desc=None*, *start_tasks=None*, *stop_tasks=None*, *do_before_data_processing=None*, *do_before_sending=None*, *valid=True*, *vtg_ids=None*)

Bases: *framework.scenario.Step*

__init__(*data_desc=None*, *final=False*, *fbk_timeout=None*, *fbk_mode=None*, *sending_delay=None*, *set_periodic=None*, *clear_periodic=None*, *step_desc=None*, *start_tasks=None*, *stop_tasks=None*, *do_before_data_processing=None*, *do_before_sending=None*, *valid=True*, *vtg_ids=None*)

Step objects are the building blocks of Scenarios.

Parameters

- **data_desc** –
- **final** –
- **fbk_timeout** –
- **fbk_mode** –
- **set_periodic** –
- **clear_periodic** –
- **step_desc** –
- **do_before_data_processing** –
- **do_before_sending** –
- **valid** –
- **vtg_ids** (*list, int*) – Virtual ID list of the targets to which the outcomes of this data process will be sent. If `None`, the outcomes will be sent to the first target that has been enabled. If `data_desc` is a list, this parameter should be a list where each item is the `vtg_ids` of the corresponding item in the `data_desc` list.
- **transition_on_dp_complete** (*bool*) – this attribute is set to `True` by the framework.
- **refresh_atoms** (*bool*) – if set to *True* atoms described by names in *data_desc* will be re-instanced each time the step is entered.

__module__ = 'framework.scenario'

class framework.scenario.**NoDataStep**(*data_desc=None*, *final=False*, *fbk_timeout=None*, *fbk_mode=None*, *sending_delay=None*, *set_periodic=None*, *clear_periodic=None*, *step_desc=None*, *start_tasks=None*, *stop_tasks=None*, *do_before_data_processing=None*, *do_before_sending=None*, *valid=True*, *vtg_ids=None*)

Bases: *framework.scenario.Step*

__annotations__ = {}

__init__(*data_desc=None*, *final=False*, *fbk_timeout=None*, *fbk_mode=None*, *sending_delay=None*, *set_periodic=None*, *clear_periodic=None*, *step_desc=None*, *start_tasks=None*, *stop_tasks=None*, *do_before_data_processing=None*, *do_before_sending=None*, *valid=True*, *vtg_ids=None*)

Step objects are the building blocks of Scenarios.

Parameters

- **data_desc** –
- **final** –
- **fbk_timeout** –
- **fbk_mode** –
- **set_periodic** –
- **clear_periodic** –
- **step_desc** –
- **do_before_data_processing** –
- **do_before_sending** –
- **valid** –
- **vtg_ids** (*list, int*) – Virtual ID list of the targets to which the outcomes of this data process will be sent. If `None`, the outcomes will be sent to the first target that has been enabled. If `data_desc` is a list, this parameter should be a list where each item is the `vtg_ids` of the corresponding item in the `data_desc` list.
- **transition_on_dp_complete** (*bool*) – this attribute is set to `True` by the framework.
- **refresh_atoms** (*bool*) – if set to *True* atoms described by names in *data_desc* will be re-instanced each time the step is entered.

__module__ = 'framework.scenario'

make_free()

class framework.scenario.**Periodic**(*data*, *period=None*, *vtg_ids=None*)

Bases: `object`

__init__(*data*, *period=None*, *vtg_ids=None*)

__module__ = 'framework.scenario'

__str__()

Return str(self).

class framework.scenario.**Scenario**(*name*, *anchor=None*, *reinit_anchor=None*, *user_context=None*, *user_args=None*)

Bases: `object`

__copy__()

__init__(*name*, *anchor=None*, *reinit_anchor=None*, *user_context=None*, *user_args=None*)
Note: only at copy the ScenarioEnv are propagated to the steps and transitions

Parameters

- **name** –
- **anchor** –
- **reinit_anchor** –
- **user_context** –
- **user_args** –

__module__ = 'framework.scenario'

__str__()
Return str(self).

_graph_setup(*init_step*, *steps*, *transitions*)

_init_main_properties()

_init_reinit_seq_properties()

_view_linux(*filepath*, *graph_filename*)
Open filepath in the user's preferred application (linux).

_view_windows(*filepath*, *graph_filename*)
Start filepath with its associated application (windows).

property anchor

branch_to_reinit(*step*, *prepend=True*)

clone(*new_name*)

property current_step

property env

graph(*fmt='pdf'*, *select_current=False*, *display_ucontext=True*)

merge_user_context_with(*user_context*)

property periodic_to_clear

property reinit_steps

property reinit_transitions

reset()

set_anchor(*anchor*, *current=None*)

set_data_model(*dm*)

set_reinit_anchor(*reinit_anchor*)

set_scenario_env(*env:* framework.scenario.ScenarioEnv, *merge_user_contexts: bool = True*)

Parameters

- **env** –

- **merge_user_contexts** – the new env will have a user_context that is the merging of the current one and the one provided through the new env. In case some parameter names overlaps, the new values are kept.

set_target(*target*)

property steps

property tasks_to_stop

property transitions

property user_context

walk_to(*step*)

walk_to_reinit()

class framework.scenario.**ScenarioEnv**

Bases: object

__copy__()

__init__()

__module__ = 'framework.scenario'

property dm

knowledge_source = None

property scenario

property target

property user_context

class framework.scenario.**Step**(*data_desc=None*, *final=False*, *fbk_timeout=None*, *fbk_mode=None*, *sending_delay=None*, *set_periodic=None*, *clear_periodic=None*, *step_desc=None*, *start_tasks=None*, *stop_tasks=None*, *do_before_data_processing=None*, *do_before_sending=None*, *valid=True*, *vtg_ids=None*, *refresh_atoms=True*)

Bases: object

__annotations__ = {}

__copy__()

__hash__()

Return hash(self).

__init__(*data_desc=None*, *final=False*, *fbk_timeout=None*, *fbk_mode=None*, *sending_delay=None*, *set_periodic=None*, *clear_periodic=None*, *step_desc=None*, *start_tasks=None*, *stop_tasks=None*, *do_before_data_processing=None*, *do_before_sending=None*, *valid=True*, *vtg_ids=None*, *refresh_atoms=True*)

Step objects are the building blocks of Scenarios.

Parameters

- **data_desc** –
- **final** –
- **fbk_timeout** –
- **fbk_mode** –

- **set_periodic** –
- **clear_periodic** –
- **step_desc** –
- **do_before_data_processing** –
- **do_before_sending** –
- **valid** –
- **vtg_ids** (*list, int*) – Virtual ID list of the targets to which the outcomes of this data process will be sent. If None, the outcomes will be sent to the first target that has been enabled. If data_desc is a list, this parameter should be a list where each item is the vtg_ids of the corresponding item in the data_desc list.
- **transition_on_dp_complete** (*bool*) – this attribute is set to True by the framework.
- **refresh_atoms** (*bool*) – if set to *True* atoms described by names in *data_desc* will be re-instanced each time the step is entered.

__module__ = 'framework.scenario'

__str__()
Return str(self).

_handle_data_desc(*data_desc*)

_stutter_cbk(*env*, *current_step*, *next_step*)

cleanup()

clear_dmaker_reset()
Restore the state changed by .set_dmaker_reset()

connect_to(*obj*, *dp_completed_guard=False*, *cbk_after_sending=None*, *cbk_after_fbk=None*, *prepend=False*, *description=None*)

property content
Provide the atom of the step if possible. In the case of a DataProcess, if it has been carried out, then the resulting atom is returned, otherwise the seed atom is returned if it exists.

Provide an atom list if the step contain multiple atom

property data_desc

do_before_data_processing()

do_before_sending()

property feedback_mode

property feedback_timeout

get_data()

get_desc(*oneliner=True*)

get_full_description(*oneliner=True*)

get_periodic_description()

get_periodic_ref()

get_tasks_description()

get_tasks_ref()

has_dataprocess()

has_tasks_to_start()

has_tasks_to_stop()

is_blocked()

is_periodic_cleared()

is_periodic_set()

make_blocked()

make_free()

make_stutter(*count=None*, *rd_count_range: Optional[Tuple[int, int]] = None*, *fbk_timeout_range: Optional[Tuple[float, float]] = None*)

Further to this call, a step is connected to itself with a guard enabling looping on the step for a number of time: either @count times or a random value within @rd_count_range.

Parameters

- **count** – number of loops.
- **rd_count_range** – number of loops is determined randomly within the bounds provided by this parameter.
- **fbk_timeout_range** – feedback timeout is chosen randomly within the bounds provided by this parameter.

property periodic_to_clear

property periodic_to_set

property sending_delay

set_dmaker_reset()

Request the framework to reset the data makers involved in the step before processing them. Relevant only when DataProcess are in use.

set_scenario_env(*env*)

set_transitions(*transitions*)

property tasks_to_start

property tasks_to_stop

property transitions

class framework.scenario.**StepStub**(*data_desc=None*, *final=False*, *fbk_timeout=None*, *fbk_mode=None*, *sending_delay=None*, *set_periodic=None*, *clear_periodic=None*, *step_desc=None*, *start_tasks=None*, *stop_tasks=None*, *do_before_data_processing=None*, *do_before_sending=None*, *valid=True*, *vtg_ids=None*)

Bases: *framework.scenario.Step*

__annotations__ = {}

__init__(*data_desc=None*, *final=False*, *fbk_timeout=None*, *fbk_mode=None*, *sending_delay=None*, *set_periodic=None*, *clear_periodic=None*, *step_desc=None*, *start_tasks=None*, *stop_tasks=None*, *do_before_data_processing=None*, *do_before_sending=None*, *valid=True*, *vtg_ids=None*)

Step objects are the building blocks of Scenarios.

Parameters

- **data_desc** –
- **final** –
- **fbk_timeout** –
- **fbk_mode** –
- **set_periodic** –
- **clear_periodic** –
- **step_desc** –
- **do_before_data_processing** –
- **do_before_sending** –
- **valid** –
- **vtg_ids** (*list, int*) – Virtual ID list of the targets to which the outcomes of this data process will be sent. If `None`, the outcomes will be sent to the first target that has been enabled. If `data_desc` is a list, this parameter should be a list where each item is the `vtg_ids` of the corresponding item in the `data_desc` list.
- **transition_on_dp_complete** (*bool*) – this attribute is set to `True` by the framework.
- **refresh_atoms** (*bool*) – if set to *True* atoms described by names in *data_desc* will be re-instanced each time the step is entered.

__module__ = 'framework.scenario'

class framework.scenario.**Transition**(*obj*, *dp_completed_guard=False*, *cbk_after_sending=None*, *cbk_after_fbk=None*, *description=None*)

Bases: `object`

__copy__()

__hash__()
Return hash(self).

__init__(*obj*, *dp_completed_guard=False*, *cbk_after_sending=None*, *cbk_after_fbk=None*, *description=None*)

__module__ = 'framework.scenario'

__str__()
Return str(self).

has_callback()

has_callback_pending()

invert_conditions()

is_crossable()

make_uncrossable()

register_callback(*callback*, *hook=HOOK.after_fbk*)

run_callback(*current_step*, *feedback=None*, *hook=HOOK.after_fbk*)

set_scenario_env(*env*, *merge_user_contexts: bool = True*)

property step

13.2.25 framework.dmhelpers.generic module

framework.dmhelpers.generic.**COPY_VALUE**(*path*, *depth=None*, *vt=None*, *set_attrs=None*, *clear_attrs=None*, *after_encoding=True*)

Return a *generator* that retrieves the value of another node, and then return a *vt* node with this value. The other node is selected:

- either directly by following the provided relative *path* from the given generator-parameter node.
- or indirectly (if *depth* is provided) where a *base* node is first selected automatically, based on our current index within our own parent node (or the nth-ancestor, depending on the parameter *depth*), and then the targeted node is selected by following the provided relative *path* from the *base* node.

Parameters

- **path** (*str*) – relative path to the node whose value will be picked.
- **depth** (*int*) – depth of our nth-ancestor used as a reference to compute automatically the targeted base node position.
- **vt** (*type*) – value type used for node generation (refer to *framework.value_types*).
- **set_attrs** (*list*) – attributes that will be set on the generated node.
- **clear_attrs** (*list*) – attributes that will be cleared on the generated node.
- **after_encoding** (*bool*) – if False, copy the raw value, otherwise the encoded one. Can be set to False only if node arguments support encoding.

framework.dmhelpers.generic.**CRC**(*vt=<class 'framework.value_types.INT_str'>*, *poly=4374732215*, *init_crc=0*, *xor_out=4294967295*, *rev=True*, *set_attrs=None*, *clear_attrs=None*, *after_encoding=True*, *freezable=False*, *base=16*, *letter_case='upper'*, *min_sz=4*, *reverse_str=False*)

Return a *generator* that returns the CRC (in the chosen type) of all the node parameters. (Default CRC is PKZIP CRC32)

Parameters

- **vt** (*type*) – value type used for node generation (refer to *framework.value_types*)
- **poly** (*int*) – CRC polynom
- **init_crc** (*int*) – initial value used to start the CRC calculation.
- **xor_out** (*int*) – final value to XOR with the calculated CRC value.
- **rev** (*bool*) – bit reversed algorithm when *True*.
- **set_attrs** (*list*) – attributes that will be set on the generated node.
- **clear_attrs** (*list*) – attributes that will be cleared on the generated node.
- **after_encoding** (*bool*) – if False compute the CRC before any encoding. Can be set to False only if node arguments support encoding.
- **freezable** (*bool*) – if `False` make the generator unfreezable in order to always provide the right value. (Note that tTYPE will still be able to corrupt the generator.)
- **base** (*int*) – Relevant when `vt` is `INT_str`. Numerical base to use for string representation
- **letter_case** (*str*) – Relevant when `vt` is `INT_str`. Letter case for string representation ('upper' or 'lower')
- **min_sz** (*int*) – Relevant when `vt` is `INT_str`. Minimum size of the resulting string.

- **reverse_str** (*bool*) – Reverse the order of the string if set to True.

framework.dmhelpers.generic.**CYCLE**(*vals*, *depth=1*, *vt=<class 'framework.value_types.String'>*, *set_attrs=None*, *clear_attrs=None*)

Return a *generator* that iterates other the provided value list and returns at each step a *vt* node corresponding to the current value.

Parameters

- **vals** (*list*) – the value list to iterate on.
- **depth** (*int*) – depth of our nth-ancestor used as a reference to iterate. By default, it is the parent node. Thus, in this case, depending on the drawn quantity of parent nodes, the position within the grand-parent determines the index of the value to use in the provided list, modulo the quantity.
- **vt** (*type*) – value type used for node generation (refer to *framework.value_types*).
- **set_attrs** (*list*) – attributes that will be set on the generated node.
- **clear_attrs** (*list*) – attributes that will be cleared on the generated node.

framework.dmhelpers.generic.**LEN**(*vt=<class 'framework.value_types.INT_str'>*, *base_len=0*, *set_attrs=None*, *clear_attrs=None*, *after_encoding=True*, *freezable=False*)

Return a *generator* that returns the length of a node parameter.

Parameters

- **vt** (*type*) – value type used for node generation (refer to *framework.value_types*).
- **base_len** (*int*) – this base length will be added to the computed length.
- **set_attrs** (*list*) – attributes that will be set on the generated node.
- **clear_attrs** (*list*) – attributes that will be cleared on the generated node.
- **after_encoding** (*bool*) – if False compute the length before any encoding. Can be set to False only if node arguments support encoding.
- **freezable** (*bool*) – If False make the generator unfreezable in order to always provide the right value. (Note that tTYPE will still be able to corrupt the generator.)

class framework.dmhelpers.generic.**MH**

Bases: object

Define constants and generator templates for data model description.

class Attr

Bases: object

Abs_Postpone = 6

DEBUG = 40

Determinist = 3

Finite = 4

Freezable = 1

Highlight = 30

LOCKED = 50

Mutable = 2

Separator = 15

```
        __module__ = 'framework.dmhelpers.generic'
    class Charset
        Bases: object
        ASCII = 1
        ASCII_EXT = 2
        UNICODE = 3
        __module__ = 'framework.dmhelpers.generic'
    Copy = 'u'
    class Custo
        Bases: object
        class Func
            Bases: object
            CloneExtNodeArgs = 2
            FrozenArgs = 1
            __module__ = 'framework.dmhelpers.generic'
        class Gen
            Bases: object
            CloneExtNodeArgs = 2
            ForwardConfChange = 1
            ResetOnUnfreeze = 3
            TriggerLast = 4
            __module__ = 'framework.dmhelpers.generic'
        class NTerm
            Bases: object
            CollapsePadding = 4
            CycleClone = 2
            DelayCollapsing = 5
            FrozenCopy = 3
            FullCombinatory = 6
            MutableClone = 1
            StickToDefault = 7
            __module__ = 'framework.dmhelpers.generic'
        __module__ = 'framework.dmhelpers.generic'
    FullyRandom = '=.'
    Generator = 2
    Leaf = 3
    NonTerminal = 1
    Ordered = '>'
```

Pick = '=+'

Random = '=..'

RawNode = 4

Regex = 5

ZeroCopy = 's'

__module__ = 'framework.dmhelpers.generic'

static _handle_attrs(*n*, *set_attrs*, *clear_attrs*)

static _validate_int_vt(*vt*)

static _validate_vt(*vt*)

framework.dmhelpers.generic.**OFFSET**(*use_current_position=True*, *depth=1*, *vt=<class 'framework.value_types.INT_str'>*, *set_attrs=None*, *clear_attrs=None*, *after_encoding=True*, *freezable=False*)

Return a *generator* that computes the offset of a child node within its parent node.

If *use_current_position* is *True*, the child node is selected automatically, based on our current index within our own parent node (or the nth-ancestor, depending on the parameter *depth*). Otherwise, the child node has to be provided in the node parameters just before its parent node.

Besides, if there are N node parameters, the first N-1 (or N-2 if *use_current_position* is False) nodes are used for adding a fixed amount (the length of their concatenated values) to the offset (determined thanks to the node in the last position of the node parameters).

The generator returns the result wrapped in a *vt* node.

Parameters

- **use_current_position** (*bool*) – automate the computation of the child node position
- **depth** (*int*) – depth of our nth-ancestor used as a reference to compute automatically the targeted child node position. Only relevant if *use_current_position* is True.
- **vt** (*type*) – value type used for node generation (refer to *framework.value_types*).
- **set_attrs** (*list*) – attributes that will be set on the generated node.
- **clear_attrs** (*list*) – attributes that will be cleared on the generated node.
- **after_encoding** (*bool*) – if False compute the fixed amount part of the offset before any encoding. Can be set to False only if node arguments support encoding.
- **freezable** (*bool*) – If `False` make the generator unfreezable in order to always provide the right value. (Note that tTYPE will still be able to corrupt the generator.)

framework.dmhelpers.generic.**QTY**(*node_name*, *vt=<class 'framework.value_types.INT_str'>*, *set_attrs=None*, *clear_attrs=None*, *freezable=False*)

Return a *generator* that returns the quantity of child node instances (referenced by name) of the node parameter provided to the *generator*.

Parameters

- **vt** (*type*) – value type used for node generation (refer to *framework.value_types*)
- **node_name** (*str*) – name of the child node whose instance amount will be returned by the generator
- **set_attrs** (*list*) – attributes that will be set on the generated node.
- **clear_attrs** (*list*) – attributes that will be cleared on the generated node.

- **freezable** (*bool*) – If `False` make the generator unfreezable in order to always provide the right value. (Note that tTYPE will still be able to corrupt the generator.)

framework.dmhelpers.generic.**SELECT**(*idx=None*, *path=None*, *filter_func=None*, *fallback_node=None*, *clone=True*, *set_attrs=None*, *clear_attrs=None*)

Return a *generator* that select a subnode from a non-terminal node and return it (or a copy of it depending on the parameter *clone*). If the *path* parameter is provided, the previous selected node is searched for the *path* in order to return the related subnode instead. The non-terminal node is selected regarding various criteria provided as parameters.

Parameters

- **idx** (*int*) – if None, the node will be selected randomly, otherwise it should be given the subnodes position in the non-terminal node, or in the subset of subnodes if the *filter_func* parameter is provided.
- **path** (*str*) – if provided, it has to be a path identifying a subnode to clone from the selected node.
- **filter_func** – function to filter the subnodes prior to any selection.
- **fallback_node** (`Node`) – if 'path' does not exist, then the clone node will be the one provided in this parameter.
- **clone** (*bool*) – [default: True] If True, the returned node will be cloned, otherwise the original node will be returned.
- **set_attrs** (*list*) – attributes that will be set on the generated node (only if *clone* is True).
- **clear_attrs** (*list*) – attributes that will be cleared on the generated node (only if *clone* is True).

framework.dmhelpers.generic.**TIMESTAMP**(*time_format='%H%M%S'*, *utc=False*, *set_attrs=None*, *clear_attrs=None*)

Return a *generator* that returns the current time (in a String node).

Parameters

- **time_format** (*str*) – time format to be used by the generator.
- **set_attrs** (*list*) – attributes that will be set on the generated node.
- **clear_attrs** (*list*) – attributes that will be cleared on the generated node.

framework.dmhelpers.generic.**WRAP**(*func*, *vt=<class 'framework.value_types.String'>*, *set_attrs=None*, *clear_attrs=None*, *after_encoding=True*, *freezable=False*)

Return a *generator* that returns the result (in the chosen type) of the provided function applied on the concatenation of all the node parameters.

Parameters

- **func** (*function*) – function applied on the concatenation
- **vt** (*type*) – value type used for node generation (refer to *framework.value_types*)
- **set_attrs** (*list*) – attributes that will be set on the generated node.
- **clear_attrs** (*list*) – attributes that will be cleared on the generated node.
- **after_encoding** (*bool*) – if False, execute *func* on node arguments before any encoding. Can be set to False only if node arguments support encoding.
- **freezable** (*bool*) – If `False` make the generator unfreezable in order to always provide the right value. (Note that tTYPE will still be able to corrupt the generator.)

13.2.26 framework.dmhelpers.xml module

class framework.dmhelpers.xml.**TAG_TYPE**(*value*)

Bases: enum.Enum

An enumeration.

__module__ = 'framework.dmhelpers.xml'

comment = 2

proc_instr = 3

standard = 1

framework.dmhelpers.xml.**tag_builder**(*tag_name*, *params=None*, *refs=None*, *contents=None*, *node_name=None*, *codec='latin-1'*, *tag_name_mutable=True*, *struct_mutable=True*, *determinist=True*, *condition=None*, *absorb_regexp=None*, *specific_fuzzy_vals=None*, *tag_type=TAG_TYPE.standard*, *nl_prefix=False*, *nl_suffix=False*)

Helper for modeling an XML tag.

Parameters

- **tag_name** (*str*) – name of the XML tag.
- **params** (*dict*) – optional attributes to be added in the XML tag
- **refs** (*dict*) – if provided it should contain for at least one parameter key (provided in params dict) the name to be used for the node representing the corresponding value. Useful when the parameter condition is in use and needs to relate to the value of specific parameters.
- **contents** – can be either None (empty tag), a framework.data_model.Node, a dictionary (Node description), a string or a string list (string-Node values).
- **node_name** (*str*) – name of the node to be created.
- **codec** (*str*) – codec to be used for generating the XML tag.
- **tag_name_mutable** (*bool*) – if False, the tag name will not be mutable, meaning that its Mutable attribute will be cleared.
- **struct_mutable** (*bool*) – if False the XML structure "will not" be mutable, meaning that each node related to the structure will have its Mutable attribute cleared.
- **determinist** (*bool*) – if False, the attribute order could change from one retrieved data to another.
- **condition** (*tuple*) – optional existence condition for the tag. If not None a keyword exists_if will be added to the root node with this parameter as a value.
- **absorb_regexp** (*str*) – regex for contents absorption
- **tag_type** (TAG_TYPE) – specify the type of notation
- **specific_fuzzy_vals** (*dict*) – if provided it should contain for at least one parameter key (provided in params dict) a list of specific values that will be used by some generic disruptors like tTYPE.
- **nl_prefix** (*bool*) – add a new line character before the tag
- **nl_suffix** (*bool*) – add a new line character after the tag

Returns Node-description of the XML tag.

Return type dict

framework.dmhelpers.xml.**xml_decl_builder**(*determinist=True*)

13.2.27 framework.evolutionary_helpers module

class framework.evolutionary_helpers.**CrossoverHelper**

Bases: object

class Operand(*node*)

Bases: object

__init__(*node*)

__module__ = 'framework.evolutionary_helpers'

_count_brothers(*index*, *pattern*)

_merge_brothers(*index*, *pattern*, *length*)

compute_sub_graphs(*percentage*)

__module__ = 'framework.evolutionary_helpers'

static _add_default_crossover_info(*ind_1*, *ind_2*, *crossover_desc=''*)

classmethod _crossover_algo2(*ind_1*, *ind_2*, *percentage_to_share*)

static _get_nodes(*node*)

static _swap_nodes(*node_1*, *node_2*)

classmethod crossover_algo1(*ind_1*, *ind_2*)

classmethod get_configured_crossover_algo2(*percentage_to_share=None*)

Parameters **percentage_to_share** – Percentage of the nodes to share.

Returns: func

class framework.evolutionary_helpers.**DefaultIndividual**(*fmk*, *data*, *mutation_order=1*)

Bases: *framework.evolutionary_helpers.Individual*

Provide a default implementation of the Individual class

__init__(*fmk*, *data*, *mutation_order=1*)

__module__ = 'framework.evolutionary_helpers'

mutate()

class framework.evolutionary_helpers.**DefaultPopulation**(*fmk*, **args*, ***kwargs*)

Bases: *framework.evolutionary_helpers.Population*

Provide a default implementation of the Population base class

__module__ = 'framework.evolutionary_helpers'

__repr__()

Return repr(self).

_compute_probability_of_survival()

Normalize fitness scores between 0 and 1

_compute_scores()
Compute the scores of each individuals

_crossover()
Compensates the kills through the usage of the COMB disruptor

_initialize(*init_process*, *max_size=100*, *max_generation_nb=50*, *crossover_algo=<bound method CrossoverHelper.crossover_algo1 of <class 'framework.evolutionary_helpers.CrossoverHelper'>>*)
Configure the population

Parameters

- **init_process** (*string*) – individuals that compose this population will be built using the provided *framework.data.DataProcess*
- **max_size** (*integer*) – maximum size of the population to manipulate
- **max_generation_nb** (*integer*) – criteria used to stop the evolution process
- **crossover_algo** (*func*) – Crossover algorithm to use

_kill()
Simply rolls the dice

_mutate()
Operates three bit flips on each individual

evolve()
Describe the evolutionary process

is_final()
Check if the population can still evolve or not

reset()
Generate the first generation of individuals in a random way

class framework.evolutionary_helpers.**EvolutionaryScenariosFactory**
Bases: object

__module__ = 'framework.evolutionary_helpers'

static build(*fmk*, *name*, *population_cls*, *args*)
Create a scenario that takes advantage of an evolutionary approach :param fmk: reference to FmkPlumbing :type fmk: FmkPlumbing :param name: name of the scenario to create :type name: string :param population_cls: population class to instantiate :type population_cls: classobj :param args (dict of str: object): arguments that will be used to instantiate a population

Returns evolutionary scenario

Return type *Scenario*

class framework.evolutionary_helpers.**Individual**(*fmk*, *data*)
Bases: object

Represents a population member

__annotations__ = {}

__init__(*fmk*, *data*)

__module__ = 'framework.evolutionary_helpers'

mutate()

class framework.evolutionary_helpers.**Population**(*fmk*, **args*, ***kwargs*)
Bases: object

Population to be used within an evolutionary scenario

__annotations__ = {}

__delitem__(*key*)

__getitem__(*key*)

__init__(*fmk*, **args*, ***kwargs*)

__iter__()

__len__()

__module__ = 'framework.evolutionary_helpers'

__next__()

__repr__()
Return repr(self).

__setitem__(*key*, *value*)

_initialize(**args*, ***kwargs*)
Initialize the population Only called once during the creating of the Population instance

evolve()
Describe the evolutionary process

is_final()
Check if the population can still evolve or not

next()

reset()
Reset the population Called before each evolutionary process

size()

13.2.28 framework.knowledge.feedback_collector module

class framework.knowledge.feedback_collector.**FeedbackCollector**
Bases: object

__init__()

__iter__()

__module__ = 'framework.knowledge.feedback_collector'

add_fbk_from(*ref*, *fbk*, *status=0*)

cleanup()

fbk_lock = <unlocked _thread.lock object>

get_bytes()

get_error_code()

get_timestamp()

has_fbk_collector()

iter_and_cleanup_collector()

set_bytes(*bstring*)

set_error_code(*err_code*)

class framework.knowledge.feedback_collector.**FeedbackSource**(*src*, *subref=None*, *reliability=None*, *related_tg=None*, *display_feedback=True*)

Bases: object

__eq__(*other*)
Return self==value.

__hash__()
Return hash(self).

__init__(*src*, *subref=None*, *reliability=None*, *related_tg=None*, *display_feedback=True*)

__module__ = 'framework.knowledge.feedback_collector'

__str__()
Return str(self).

property display_feedback

property obj

property related_tg

13.2.29 framework.knowledge.feedback_handler module

class framework.knowledge.feedback_handler.**FeedbackHandler**(*new_window=False*, *new_window_title=None*)

Bases: object

A feedback handler extract information from binary data.

__init__(*new_window=False*, *new_window_title=None*)

Parameters **new_window** – If *True*, a new terminal emulator is created, enabling the decoder to use it for display via the methods *print()* and *print_nl()*

__module__ = 'framework.knowledge.feedback_handler'

_start(*current_dm*)

_stop()

collect_data(*s*)

estimate_last_data_impact_uniqueness()
*** To be overloaded ***

Estimate the similarity of the consequences triggered by the current data sending from previous sending. Estimation can be computed with provided feedback.

Returns provide an estimation of impact similarity

Return type *SimilarityMeasure*

extract_info_from_feedback(*current_dm*, *source*, *timestamp*, *content*, *status*)
*** To be overloaded ***

Parameters

- **current_dm** (*framework.data_model.DataModel*) – current loaded DataModel
- **source** (*framework.knowledge.feedback_collector.FeedbackSource*) – source of the feedback
- **timestamp** (*datetime*) – date of reception of the feedback
- **content** (*bytes*) – binary data to process
- **status** (*int*) – negative status signify an error

Returns a set of *information.Info* or only one

Return type *Info*

flush_collector()

notify_data_sending(*current_dm*, *data_list*, *timestamp*, *target*)

*** To be overloaded ***

This function is called when data have been sent. It enables to process feedback relatively to previously sent data.

Parameters

- **current_dm** (*framework.data_model.DataModel*) – current loaded DataModel
- **data_list** (*list*) – list of *framework.data.Data* that were sent
- **timestamp** (*datetime*) – date when data was sent
- **target** (*framework.target_helpers.Target*) – target to which data was sent

print(*msg*)

print_nl(*msg*)

process_feedback(*current_dm*, *source*, *timestamp*, *content*, *status*)

start(*current_dm*)

stop()

class framework.knowledge.feedback_handler.**SimilarityMeasure**(*level=0*)

Bases: object

__add__(*other*)

__eq__(*other*)

Return self==value.

__ge__(*other*, *NotImplemented=NotImplemented*)

Return a >= b. Computed by @total_ordering from (not a < b).

__gt__(*other*, *NotImplemented=NotImplemented*)

Return a > b. Computed by @total_ordering from (not a < b) and (a != b).

__hash__ = None

__init__(*level=0*)

__le__(*other*, *NotImplemented=NotImplemented*)

Return a <= b. Computed by @total_ordering from (a < b) or (a == b).

__lt__(*other*)

Return self<value.

__module__ = 'framework.knowledge.feedback_handler'

property value

class framework.knowledge.feedback_handler.**TestFbkHandler**(*new_window=False*, *new_window_title=None*)

Bases: *framework.knowledge.feedback_handler.FeedbackHandler*

__annotations__ = {}

__module__ = 'framework.knowledge.feedback_handler'

extract_info_from_feedback(*current_dm*, *source*, *timestamp*, *content*, *status*)

* To be overloaded *

Parameters

- **current_dm** (*framework.data_model.DataModel*) – current loaded DataModel
- **source** (*framework.knowledge.feedback_collector.FeedbackSource*) – source of the feedback
- **timestamp** (*datetime*) – date of reception of the feedback
- **content** (*bytes*) – binary data to process
- **status** (*int*) – negative status signify an error

Returns a set of *information.Info* or only one

Return type *Info*

13.2.30 framework.knowledge.information module

class framework.knowledge.information.**Hardware**(*value*)

Bases: *framework.knowledge.information.Info*

An enumeration.

ARM = 4

PowerPc = 3

Unknown = 5

X86_32 = 2

X86_64 = 1

__module__ = 'framework.knowledge.information'

class framework.knowledge.information.**Info**(*value*)

Bases: enum.Enum

An enumeration.

__init__(*val*)

__module__ = 'framework.knowledge.information'

decrease_trust(*inc=1*)

increase_trust(*inc=1*)

reset_trust()

property trust_level

property trust_value

class framework.knowledge.information.InformationCollector
Bases: object

__bool__()

__init__()

__module__ = 'framework.knowledge.information'

__str__()
Return str(self).

add_information(*info*, *initial_trust_value=0*)

is_assumption_valid(*info*)

is_info_class_represented(*info_class*)

reset_information()

class framework.knowledge.information.InputHandling(*value*)
Bases: *framework.knowledge.information.Info*

An enumeration.

Ctrl_Char_Set = 1

Printable_Char_Set = 2

Unknown = 3

__module__ = 'framework.knowledge.information'

class framework.knowledge.information.Language(*value*)
Bases: *framework.knowledge.information.Info*

An enumeration.

Ada = 2

C = 1

Pascal = 3

Unknown = 4

__module__ = 'framework.knowledge.information'

class framework.knowledge.information.OS(*value*)
Bases: *framework.knowledge.information.Info*

An enumeration.

Android = 3

Linux = 1

Unknown = 4

Windows = 2

__module__ = 'framework.knowledge.information'

class framework.knowledge.information.OperationMode(*value*)
Bases: *framework.knowledge.information.Info*

An enumeration.

Determinist = 1

Random = 2

__module__ = 'framework.knowledge.information'

class framework.knowledge.information.**Test**(*value*)

Bases: *framework.knowledge.information.Info*

An enumeration.

Cursory = 1

Deep = 3

Medium = 2

__module__ = 'framework.knowledge.information'

class framework.knowledge.information.**TrustLevel**(*value*)

Bases: enum.Enum

An enumeration.

Maximum = 1

Medium = 2

Minimum = 3

__module__ = 'framework.knowledge.information'

13.2.31 framework.constraint_helpers module

class framework.constraint_helpers.**CSP**(*constraints: Optional[*framework.constraint_helpers.Constraint*] = None*, *highlight_variables=False*)

Bases: object

__copy__()

__init__(*constraints: Optional[*framework.constraint_helpers.Constraint*] = None*, *highlight_variables=False*)

__module__ = 'framework.constraint_helpers'

_constraints = None

_exhausted_solutions = None

_is_solution_queried = False

_model = None

_orig_var_domain = None

_problem = None

_solutions = None

_solve_constraints()

_var_domain = None

_var_domain_updated = False

_var_node_mapping = None

_var_to_varns = None

_vars = None

property exhausted_solutions

from_var_to_varns(*var*)

get_all_constraints()

get_constraint(*idx*)

get_solution()

highlight_variables = None

property is_current_solution_queried

iter_vars()

map_var_to_node(*var*, *node*)

property nb_constraints

negate_constraint(*idx*)

next_solution()

reset()

reset_constraint(*idx*)

restore_var_domains()

save_current_var_domains()

set_var_domain(*var*, *domain*)

property var_domain_updated

property var_mapping

class framework.constraint_helpers.**Constraint**(*relation*, *vars: Tuple*, *var_to_varns: Optional[dict] = None*)

Bases: object

__copy__()

__init__(*relation*, *vars: Tuple*, *var_to_varns: Optional[dict] = None*)

Parameters

- **relation** – boolean function that define the constraints between variables
- **vars** (*list*) – list of the names of the nodes used in the boolean function in *relation* (in the same order as the parameters of the function).
- **var_to_varns** (*dict*) – dictionary that associates for each name in *vars*, the comprehensive reference to the related node, which is a tuple of its name and its namespace.

__module__ = 'framework.constraint_helpers'

_negated_relation(**args*)

_orig_relation = None

_var_domain = None

negate()

relation = None

reset_to_original()

property var_domain

vars = None

exception framework.constraint_helpers.**ConstraintError**
Bases: Exception

__module__ = 'framework.constraint_helpers'

13.2.32 framework.plumbing module

class framework.plumbing.**ExportableFMKOps**(*fmk*)

class framework.plumbing.**FmkPlumbing**(*exit_on_error=False*, *debug_mode=False*, *quiet=False*, *external_term=False*, *fmkdb_path=None*)
Defines the methods to operate every sub-systems of fuddly

_delay_sending()
return False if the user want to stop fuzzing (action possible if delay is set to -1)

_log_directly_retrieved_target_feedback(*tg*, *preamble=None*, *epilogue=None*)
This method is to be used when the target does not make use of Logger.collect_feedback() facility. We thus try to access the feedback from Target directly

_send_data(*data_list: Sequence[*framework.data.Data*]*)
@data_list: either a list of Data() or a Data()

cleanup_all_dmakers(*reset_existing_seed=True*)

cleanup_dmaker(*dmaker_type=None*, *name=None*, *dmaker_obj=None*, *reset_existing_seed=True*, *error_on_init=True*)

collect_residual_feedback(*timeout=0*)

disable_fmkdb()

disable_wkspace()

display_color_theme()

dynamic_generator_ids()

empty_data_bank()

empty_workspace()

enable_fmkdb()

enable_wkspace()

exec_dm_tests()

flush_errors()

fmkdb_fetch_data(*start_id=1*, *end_id=- 1*)

get_available_targets()

get_data_model_by_name(*name*)

get_data_models(*fmkDB_update=True*)

get_error()

get_from_data_bank(*i*)

get_last_data()

get_operator(*name*)

get_probe_delay(*name*)

get_project_by_name(*name*)

get_projects(*fmkDB_update=True*)

handle_data_desc(*data_desc*, *resolve_dataprocess=True*, *original_data=None*, *save_generator_seed=False*, *reset_dmakers=False*)

is_not_ok()

is_ok()

is_target_enabled()

is_usable()

iter_data_bank()

iter_data_models()

launch()

launch_operator(*name*, *user_input=None*, *use_existing_seed=True*, *verbose=False*)

launch_probe(*name*)

load_data_model(*dm=None*, *name=None*)

load_multiple_data_model(*dm_list=None*, *name_list=None*, *reload_dm=False*)

load_project(*prj=None*, *name=None*)

load_targets(*tg_ids*)

log_comment(*comments*)

log_target_residual_feedback()

monitor_probes(*prefix=None*, *force_record=False*)

property prj

process_data(*action_list*, *seed=None*, *valid_gen=False*, *save_gen_seed=False*, *reset_dmakers=False*)

Parameters **action_list** (*list*) – Shall have a format compatible with what follows [(action_1, UserInput_1), …, (action_n, UserInput_n)] [action_1, (action_2, UserInput_2), … action_n] where action_N can be either: dmaker_type_N or (dmaker_type_N, dmaker_name_N)

process_data_and_send(*data_desc=None*, *id_from_fmkdb=None*, *id_from_db=None*, *max_loop=1*, *tg_ids=None*, *verbose=False*, *console_display=True*, *save_generator_seed=False*)

Send data to the selected targets. These data can follow a specific processing before being emitted. The latter depends on what is provided in *data_desc*.

Parameters

- **data_desc** – Can be either a *framework.data.DataProcess*, a *framework.data.Data*, the name (str) of an atom of the loaded data models, or a list of the previous types.

- **id_from_fmkdb** – Data can be fetched from the FmkDB and send directly to the targets or be used as the seed of a DataProcess if such object is provided in *data_desc*.
- **id_from_db** – Data can be fetched from the Data Bank and send directly to the targets or be used as the seed of a DataProcess if such object is provided in *data_desc*.
- **max_loop** – Maximum number of iteration. -1 one means "infinite" or until some criteria occurs (e.g., a disruptor has exhausted, the end-user issued Ctrl-C, ...)
- **tg_ids** – Target ID or list of the Target IDs on which data will be sent. If provided it will supersede the *tg_ids* parameter of any DataProcess provided in *data_desc*
- **verbose** – Pretty print sent data
- **console_display** – If *False*, nothing will be displayed on the screen (that could cause latency)
- **save_generator_seed** – If random Generators are used, the generated data will be internally saved and will be reused next time this generator will be called, until FmkPlumbing.cleanup_dmaker(... reset_existing_seed=True) is called on this Generator.

Returns The list of data that have been sent. *None* if nothing was sent due to some error.

projects()

register_current_in_data_bank()

register_in_data_bank(*data*)

register_last_in_data_bank()

reload_all(*tg_ids=None*)

reload_dm()

retrieve_and_log_target_feedback(*residual=False*)

run_project(*prj=None*, *name=None*, *tg_ids=None*, *dm_name=None*)

send_data_and_log(*data_list*, *verbose=False*, *console_display=True*)

set_disruptor_weight(*dmaker_type*, *data_maker_name*, *weight*)

set_error(*msg=''*, *context=None*, *code=- 1*)

set_feedback_mode(*mode*, *tg_id=None*, *do_record=False*, *do_show=True*)

set_feedback_timeout(*timeout*, *tg_id=None*, *do_record=True*, *do_show=True*)

set_generator_weight(*generator_type*, *data_maker_name*, *weight*)

set_health_check_timeout(*timeout*, *target=None*, *do_record=True*, *do_show=True*)

set_probe_delay(*name*, *delay*)

set_sending_burst_counter(*val*, *do_record=False*)

set_sending_delay(*delay*, *do_record=True*)

show_and_flush_errors()

show_atom_identifiers()

show_data(*data:* framework.data.Data, *verbose=True*)

show_data_bank()

show_data_maker_types()

show_data_models()

show_disruptors(*dmaker_type=None*)

show_fmk_internals()

show_generators(*dmaker_type=None*)

show_knowledge()

show_operators()

show_probes()

show_projects()

show_scenario(*sc_name*, *fmt='pdf'*)

show_targets()

show_tasks()

show_wkspace()

start()

stop()

stop_all_probes()

stop_all_tasks()

stop_probe(*name*)

switch_feedback_mode(*tg_id*, *do_record=False*, *do_show=True*)

switch_term()

wait_for_target_readiness(*forced_feedback_timeout=None*)

Parameters **forced_feedback_timeout** – should be an integer >= 0 if only feedback need to be checked

Returns:

class framework.plumbing.**FmkTask**(*name*, *func*, *arg*, *period=None*, *error_func=<function FmkTask.<lambda>>*, *cleanup_func=<function FmkTask.<lambda>>*)

run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

stop()

class framework.plumbing.**Printer**(*fmk*)

flush()

Flush write buffers, if applicable.

This is not implemented for read-only and non-blocking streams.

print(*msg*)

start()

stop()

wait_for_sync()

write(*data*)
Write string to file.

Returns the number of characters written, which is always equal to the length of the string.

13.2.33 libs.utils module

class libs.utils.**Accumulator**
Bases: object

accumulate(*msg*)

clear()

class libs.utils.**ExternalDisplay**
Bases: object

property disp

property is_enabled

property is_terminal

start_term(*title=None*, *keepterm=False*)

stop()

class libs.utils.**Task**(*period=None*, *init_delay=0*, *new_window=False*, *new_window_title=None*)
Bases: object

cleanup()

dm = None

feedback_gate = None

fmkops = None

period = None

print(*msg*)

print_nl(*msg*)

prj = None

setup()

targets = None

class libs.utils.**Term**(*title=None*, *keepterm=False*)
Bases: object

print(*s*, *newline=False*)

print_nl(*s*)

start()

stop()

libs.utils.**chunk_lines**(*string*, *length*, *prefix=''*)

libs.utils.**find_file**(*filename*, *root_path*)

libs.utils.**get_caller_object**(*stack_frame=2*)

libs.utils.**retrieve_app_handler**(*filename*)

CHAPTER

FOURTEEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

f

framework.basic_primitives, 155
framework.comm_backends, 257
framework.constraint_helpers, 296
framework.data, 155
framework.data_model, 160
framework.database, 273
framework.dmhelpers.generic, 283
framework.dmhelpers.xml, 288
framework.encoders, 271
framework.evolutionary_helpers, 289
framework.fuzzing_primitives, 266
framework.generic_data_makers, 219
framework.knowledge.feedback_collector, 291
framework.knowledge.feedback_handler, 292
framework.knowledge.information, 294
framework.logger, 247
framework.monitor, 249
framework.node, 163
framework.node_builder, 190
framework.operator_helpers, 245
framework.plumbing, 298
framework.project, 244
framework.scenario, 276
framework.tactics_helpers, 261
framework.target_helpers, 229
framework.targets.debug, 242
framework.targets.local, 238
framework.targets.network, 233
framework.targets.printer, 241
framework.targets.ssh, 239
framework.value_types, 200

l

libs.utils, 302

INDEX

Symbols

_INTERNALS_ID (*framework.targets.network.NetworkTarget attribute*), 233

__add__() (*framework.knowledge.feedback_handler.SimilarityMeasure method*), 293

__annotations__ (*framework.comm_backends.SSH_Backend attribute*), 258

__annotations__ (*framework.comm_backends.Serial_Backend attribute*), 260

__annotations__ (*framework.comm_backends.Shell_Backend attribute*), 261

__annotations__ (*framework.data.DataAttr attribute*), 158

__annotations__ (*framework.data.EmptyBackend attribute*), 159

__annotations__ (*framework.data.NodeBackend attribute*), 159

__annotations__ (*framework.data.RawBackend attribute*), 160

__annotations__ (*framework.encoders.Encoder attribute*), 271

__annotations__ (*framework.encoders.GSM7bitPacking_Enc attribute*), 272

__annotations__ (*framework.encoders.GSMPhoneNum_Enc attribute*), 272

__annotations__ (*framework.encoders.GZIP_Enc attribute*), 272

__annotations__ (*framework.encoders.Wrap_Enc attribute*), 273

__annotations__ (*framework.evolutionary_helpers.Individual attribute*), 290

__annotations__ (*framework.evolutionary_helpers.Population attribute*), 291

__annotations__ (*framework.fuzzing_primitives.BasicVisitor attribute*), 267

__annotations__ (*framework.fuzzing_primitives.NodeConsumerStub attribute*), 268

__annotations__ (*framework.fuzzing_primitives.NonTermVisitor attribute*), 269

__annotations__ (*framework.fuzzing_primitives.SeparatorDisruption attribute*), 270

__annotations__ (*framework.fuzzing_primitives.TypedNodeDisruption attribute*), 270

__annotations__ (*framework.generic_data_makers.d_call_external_program attribute*), 219

__annotations__ (*framework.generic_data_makers.d_call_function attribute*), 220

__annotations__ (*framework.generic_data_makers.d_corrupt_bits_by_position attribute*), 220

__annotations__ (*framework.generic_data_makers.d_corrupt_node_bits attribute*), 220

__annotations__ (*framework.generic_data_makers.d_fix_constraints attribute*), 221

__annotations__ (*framework.generic_data_makers.d_fuzz_model_structure attribute*), 221

__annotations__ (*framework.generic_data_makers.d_max_size attribute*), 221

__annotations__ (*framework.generic_data_makers.d_modify_nodes attribute*), 222

__annotations__ (*framework.generic_data_makers.d_next_node_content attribute*), 222

__annotations__ (*frame-*

work.generic_data_makers.d_operate_on_nodes attribute), 223
__annotations__ (framework.generic_data_makers.d_shallow_copy attribute), 223
__annotations__ (framework.generic_data_makers.d_switch_to_alternate_conf attribute), 223
__annotations__ (framework.generic_data_makers.g_generic_pattern attribute), 224
__annotations__ (framework.generic_data_makers.g_population attribute), 224
__annotations__ (framework.generic_data_makers.sd_constraint_fuzz attribute), 224
__annotations__ (framework.generic_data_makers.sd_fuzz_separator_nodes attribute), 225
__annotations__ (framework.generic_data_makers.sd_fuzz_typed_nodes attribute), 226
__annotations__ (framework.generic_data_makers.sd_struct_constraints attribute), 227
__annotations__ (framework.generic_data_makers.sd_switch_to_alternate_conf attribute), 227
__annotations__ (framework.generic_data_makers.sd_walk_csp_solutions attribute), 228
__annotations__ (framework.generic_data_makers.sd_walk_data_model attribute), 228
__annotations__ (framework.knowledge.feedback_handler.TestFbkHandler attribute), 294
__annotations__ (framework.monitor.ProbeCmd attribute), 252
__annotations__ (framework.monitor.ProbeMem attribute), 254
__annotations__ (framework.monitor.ProbePID attribute), 255
__annotations__ (framework.monitor.ProbeUser attribute), 256
__annotations__ (framework.node.GenFuncCusto attribute), 166
__annotations__ (framework.node.IntCondition attribute), 166
__annotations__ (framework.node.NodeCondition attribute), 175
__annotations__ (framework.node.NodeCustomization attribute), 176
__annotations__ (framework.node.NodeInternals_Empty attribute), 178
__annotations__ (framework.node.NodeInternals_Func attribute), 178
__annotations__ (framework.node.NodeInternals_GenFunc attribute), 179
__annotations__ (framework.node.NodeInternals_NonTerm attribute), 181
__annotations__ (framework.node.NodeInternals_Term attribute), 184
__annotations__ (framework.node.NodeInternals_TypedValue attribute), 185
__annotations__ (framework.node.NonTermCusto attribute), 188
__annotations__ (framework.node.RawCondition attribute), 188
__annotations__ (framework.node.SyncObj attribute), 189
__annotations__ (framework.node.SyncQtyFromObj attribute), 189
__annotations__ (framework.node.SyncSizeObj attribute), 190
__annotations__ (framework.node_builder.RegexParser.Brackets attribute), 193
__annotations__ (framework.node_builder.RegexParser.Brackets.Comma attribute), 192
__annotations__ (framework.node_builder.RegexParser.Brackets.Final attribute), 192
__annotations__ (framework.node_builder.RegexParser.Brackets.Initial attribute), 192
__annotations__ (framework.node_builder.RegexParser.Brackets.Max attribute), 193
__annotations__ (framework.node_builder.RegexParser.Brackets.Min attribute), 193
__annotations__ (framework.node_builder.RegexParser.Choice attribute), 193
__annotations__ (framework.node_builder.RegexParser.Dot attribute), 193
__annotations__ (framework.node_builder.RegexParser.Escape at-

tribute), 194
__annotations__ (*framework.node_builder.RegexParser.EscapeMetaSequence attribute*), 194
__annotations__ (*framework.node_builder.RegexParser.Final attribute*), 194
__annotations__ (*framework.node_builder.RegexParser.Group attribute*), 194
__annotations__ (*framework.node_builder.RegexParser.Initial attribute*), 194
__annotations__ (*framework.node_builder.RegexParser.Main attribute*), 195
__annotations__ (*framework.node_builder.RegexParser.Parenthesis attribute*), 196
__annotations__ (*framework.node_builder.RegexParser.Parenthesis.Choice attribute*), 195
__annotations__ (*framework.node_builder.RegexParser.Parenthesis.Escape attribute*), 195
__annotations__ (*framework.node_builder.RegexParser.Parenthesis.Final attribute*), 195
__annotations__ (*framework.node_builder.RegexParser.Parenthesis.Initial attribute*), 196
__annotations__ (*framework.node_builder.RegexParser.Parenthesis.Main attribute*), 196
__annotations__ (*framework.node_builder.RegexParser.QtyState attribute*), 196
__annotations__ (*framework.node_builder.RegexParser.SquareBrackets attribute*), 198
__annotations__ (*framework.node_builder.RegexParser.SquareBrackets.AfterRange attribute*), 197
__annotations__ (*framework.node_builder.RegexParser.SquareBrackets.BeforeRange attribute*), 197
__annotations__ (*framework.node_builder.RegexParser.SquareBrackets.EscapeAfterRange attribute*), 197
__annotations__ (*framework.node_builder.RegexParser.SquareBrackets.EscapeBeforeRange attribute*), 197
__annotations__ (*framework.node_builder.RegexParser.SquareBrackets.EscapeMetaSequence attribute*), 198
__annotations__ (*framework.node_builder.RegexParser.SquareBrackets.Final attribute*), 198
__annotations__ (*framework.node_builder.RegexParser.SquareBrackets.Initial attribute*), 198
__annotations__ (*framework.node_builder.RegexParser.SquareBrackets.Range attribute*), 198
__annotations__ (*framework.node_builder.State attribute*), 199
__annotations__ (*framework.node_builder.StateMachine attribute*), 199
__annotations__ (*framework.scenario.NoDataStep attribute*), 277
__annotations__ (*framework.scenario.Step attribute*), 279
__annotations__ (*framework.scenario.StepStub attribute*), 281
__annotations__ (*framework.tactics_helpers.DataMaker attribute*), 261
__annotations__ (*framework.tactics_helpers.Disruptor attribute*), 262
__annotations__ (*framework.tactics_helpers.DynGenerator attribute*), 262
__annotations__ (*framework.tactics_helpers.DynGeneratorFromScenario attribute*), 263
__annotations__ (*framework.tactics_helpers.Generator attribute*), 264
__annotations__ (*framework.tactics_helpers.StatefulDisruptor attribute*), 264
__annotations__ (*framework.tactics_helpers.dyn_generator attribute*), 266
__annotations__ (*framework.tactics_helpers.dyn_generator_from_scenario attribute*), 266
__annotations__ (*framework.target_helpers.Target attribute*), 230
__annotations__ (*framework.targets.debug.TestTarget attribute*), 242
__annotations__ (*framework.targets.local.LocalTarget attribute*), 238
__annotations__ (*framework.targets.network.NetworkTarget attribute*), 252

__annotations__ (*framework.targets.printer.PrinterTarget attribute*), 241
__annotations__ (*framework.targets.ssh.SSHTarget attribute*), 239
__annotations__ (*framework.value_types.Filename attribute*), 202
__annotations__ (*framework.value_types.FolderPath attribute*), 204
__annotations__ (*framework.value_types.GSM7bitPacking attribute*), 205
__annotations__ (*framework.value_types.GSMPhoneNum attribute*), 205
__annotations__ (*framework.value_types.GZIP attribute*), 205
__annotations__ (*framework.value_types.INT attribute*), 206
__annotations__ (*framework.value_types.INT16 attribute*), 207
__annotations__ (*framework.value_types.INT32 attribute*), 207
__annotations__ (*framework.value_types.INT64 attribute*), 207
__annotations__ (*framework.value_types.INT8 attribute*), 208
__annotations__ (*framework.value_types.INT_str attribute*), 208
__annotations__ (*framework.value_types.SINT16_be attribute*), 209
__annotations__ (*framework.value_types.SINT16_le attribute*), 209
__annotations__ (*framework.value_types.SINT32_be attribute*), 209
__annotations__ (*framework.value_types.SINT32_le attribute*), 209
__annotations__ (*framework.value_types.SINT64_be attribute*), 210
__annotations__ (*framework.value_types.SINT64_le attribute*), 210
__annotations__ (*framework.value_types.SINT8 attribute*), 210
__annotations__ (*framework.value_types.String attribute*), 211
__annotations__ (*framework.value_types.UINT16_be attribute*), 215
__annotations__ (*framework.value_types.UINT16_le attribute*), 215
__annotations__ (*framework.value_types.UINT32_be attribute*), 216
__annotations__ (*framework.value_types.UINT32_le attribute*), 216
__annotations__ (*framework.value_types.UINT64_be attribute*), 216
__annotations__ (*framework.value_types.UINT64_le attribute*), 217
__annotations__ (*framework.value_types.UINT8 attribute*), 217
__annotations__ (*framework.value_types.VT attribute*), 217
__annotations__ (*framework.value_types.VT_Alt attribute*), 218
__annotations__ (*framework.value_types.Wrapper attribute*), 218
__bool__() (*framework.database.FeedbackGate method*), 275
__bool__() (*framework.knowledge.information.InformationCollector method*), 295
__bool__() (*framework.node.NodeSemanticsCriteria method*), 187
__clear_dmaker_clones() (*framework.tactics_helpers.Tactics method*), 265
__clone_dmaker() (*framework.tactics_helpers.Tactics method*), 265
__compute_total_possible_values() (*framework.value_types.BitField method*), 200
__copy__() (*framework.constraint_helpers.CSP method*), 296
__copy__() (*framework.constraint_helpers.Constraint method*), 297
__copy__() (*framework.data.AttrGroup method*), 155
__copy__() (*framework.data.Data method*), 156
__copy__() (*framework.data.DataProcess method*), 158
__copy__() (*framework.data.NodeBackend method*), 159
__copy__() (*framework.encoders.Encoder method*), 271
__copy__() (*framework.node.DynNode_Helpers method*), 163
__copy__() (*framework.node.Env method*), 164
__copy__() (*framework.node.Env4NT method*), 165
__copy__() (*framework.node.Node method*), 167
__copy__() (*framework.node.NodeCustomization method*), 176
__copy__() (*framework.node.NodeInternals_NonTerm.NodeAttrs method*), 181
__copy__() (*framework.scenario.Scenario method*), 277
__copy__() (*framework.scenario.ScenarioEnv method*), 279
__copy__() (*framework.scenario.Step method*), 279
__copy__() (*framework.scenario.Transition method*), 282
__delitem__() (*framework.evolutionary_helpers.Population method*), 291
__eq__() (*framework.knowledge.feedback_collector.FeedbackSource method*), 292
__eq__() (*framework.knowledge.feedback_handler.SimilarityMeasure

method), 293
__ge__() (*framework.knowledge.feedback_handler.SimilarityMeasure method*), 293
__get_confs() (*framework.node.Node method*), 167
__get_current_internals() (*framework.node.Node method*), 167
__get_internals() (*framework.node.Node method*), 168
__get_node_from_db() (*framework.node_builder.NodeBuilder method*), 190
__get_random_data_maker() (*framework.tactics_helpers.Tactics method*), 265
__get_value_specific_mode1() (*framework.node.NodeInternals_Func method*), 178
__get_value_specific_mode2() (*framework.node.NodeInternals_Func method*), 179
__getattr__() (*framework.node.Env method*), 164
__getattr__() (*framework.node.Node method*), 168
__getattr__() (*framework.node.NodeInternals_GenFunc method*), 179
__getattr__() (*framework.node.NodeInternals_TypedValue method*), 185
__getitem__() (*framework.evolutionary_helpers.Population method*), 291
__getitem__() (*framework.node.Node method*), 168
__getitem__() (*framework.node.NodeCustomization method*), 176
__gt__() (*framework.knowledge.feedback_handler.SimilarityMeasure method*), 293
__handle_clone() (*framework.node_builder.NodeBuilder method*), 190
__handle_transition_callbacks() (*framework.tactics_helpers.DynGeneratorFromScenario method*), 263
__hash__ (*framework.knowledge.feedback_handler.SimilarityMeasure attribute*), 293
__hash__() (*framework.knowledge.feedback_collector.FeedbackSource method*), 292
__hash__() (*framework.node.Node method*), 168
__hash__() (*framework.node.NodeInternals method*), 176
__hash__() (*framework.scenario.Step method*), 279
__hash__() (*framework.scenario.Transition method*), 282
__id__() (*framework.node.DJobGroup method*), 163
__init__() (*framework.comm_backends.Backend method*), 257
__init__() (*framework.comm_backends.BackendError method*), 258
__init__() (*framework.comm_backends.SSH_Backend method*), 258
__init__() (*framework.comm_backends.Serial_Backend method*), 260
__init__() (*framework.comm_backends.Shell_Backend method*), 261
__init__() (*framework.constraint_helpers.CSP method*), 296
__init__() (*framework.constraint_helpers.Constraint method*), 297
__init__() (*framework.data.AttrGroup method*), 155
__init__() (*framework.data.CallBackOps method*), 156
__init__() (*framework.data.Data method*), 156
__init__() (*framework.data.DataAttr method*), 158
__init__() (*framework.data.DataBackend method*), 158
__init__() (*framework.data.DataProcess method*), 158
__init__() (*framework.data.EmptyDataProcess method*), 159
__init__() (*framework.data_model.DataModel method*), 160
__init__() (*framework.data_model.NodeBackend method*), 163
__init__() (*framework.database.Database method*), 273
__init__() (*framework.database.FeedbackGate method*), 275
__init__() (*framework.encoders.Encoder method*), 271
__init__() (*framework.evolutionary_helpers.CrossoverHelper.Operand method*), 289
__init__() (*framework.evolutionary_helpers.DefaultIndividual method*), 289
__init__() (*framework.evolutionary_helpers.Individual method*), 290
__init__() (*framework.evolutionary_helpers.Population method*), 291
__init__() (*framework.fuzzing_primitives.ModelWalker method*), 268
__init__() (*framework.fuzzing_primitives.NodeConsumerStub method*), 268
__init__() (*framework.knowledge.feedback_collector.FeedbackCollector method*), 291
__init__() (*framework.knowledge.feedback_collector.FeedbackSource method*), 292
__init__() (*framework.knowledge.feedback_handler.FeedbackHandler method*), 292
__init__() (*framework.knowledge.feedback_handler.SimilarityMeasure method*), 293
__init__() (*framework.knowledge.information.Info method*), 294
__init__() (*framework.knowledge.information.InformationCollector*

method), 295
__init__() (framework.logger.Logger method), 247
__init__() (framework.monitor.AddExistingProbeToMonitorError method), 249
__init__() (framework.monitor.BlockingProbeUser method), 249
__init__() (framework.monitor.Monitor method), 250
__init__() (framework.monitor.Probe method), 251
__init__() (framework.monitor.ProbeCmd method), 252
__init__() (framework.monitor.ProbeMem method), 254
__init__() (framework.monitor.ProbePID method), 255
__init__() (framework.monitor.ProbeStatus method), 256
__init__() (framework.monitor.ProbeTimeoutError method), 256
__init__() (framework.monitor.ProbeUser method), 256
__init__() (framework.node.BitFieldCondition method), 163
__init__() (framework.node.DJobGroup method), 163
__init__() (framework.node.DynNode_Helpers method), 164
__init__() (framework.node.Env method), 164
__init__() (framework.node.Env4NT method), 165
__init__() (framework.node.IntCondition method), 166
__init__() (framework.node.Node method), 168
__init__() (framework.node.NodeCustomization method), 176
__init__() (framework.node.NodeInternals method), 176
__init__() (framework.node.NodeInternalsCriteria method), 178
__init__() (framework.node.NodeSemantics method), 186
__init__() (framework.node.NodeSemanticsCriteria method), 187
__init__() (framework.node.NodeSeparator method), 187
__init__() (framework.node.RawCondition method), 188
__init__() (framework.node.SyncExistenceObj method), 188
__init__() (framework.node.SyncQtyFromObj method), 189
__init__() (framework.node.SyncSizeObj method), 190
__init__() (framework.node_builder.NodeBuilder method), 190
__init__() (framework.node_builder.State method), 199
__init__() (framework.node_builder.StateMachine method), 199
__init__() (framework.operator_helpers.LastInstruction method), 245
__init__() (framework.operator_helpers.Operation method), 246
__init__() (framework.project.Project method), 244
__init__() (framework.scenario.FinalStep method), 276
__init__() (framework.scenario.NoDataStep method), 277
__init__() (framework.scenario.Periodic method), 277
__init__() (framework.scenario.Scenario method), 277
__init__() (framework.scenario.ScenarioEnv method), 279
__init__() (framework.scenario.Step method), 279
__init__() (framework.scenario.StepStub method), 281
__init__() (framework.scenario.Transition method), 282
__init__() (framework.tactics_helpers.DataMaker method), 261
__init__() (framework.tactics_helpers.Disruptor method), 262
__init__() (framework.tactics_helpers.Generator method), 264
__init__() (framework.tactics_helpers.StatefulDisruptor method), 264
__init__() (framework.tactics_helpers.Tactics method), 265
__init__() (framework.tactics_helpers.dyn_generator method), 266
__init__() (framework.target_helpers.EmptyTarget method), 229
__init__() (framework.target_helpers.Target method), 230
__init__() (framework.targets.debug.TestTarget method), 242
__init__() (framework.targets.local.LocalTarget method), 238
__init__() (framework.targets.network.NetworkTarget method), 233
__init__() (framework.targets.printer.PrinterTarget method), 241
__init__() (framework.targets.ssh.SSHTarget method), 239
__init__() (framework.value_types.BitField method), 200
__init__() (framework.value_types.INT method), 206
__init__() (framework.value_types.INT_str method), 208
__init__() (framework.value_types.String method), 211
__init__() (framework.value_types.VT_Alt method), 218
__iter__() (framework.database.FeedbackGate

method), 275
__iter__() (*framework.evolutionary_helpers.Population method*), 291
__iter__() (*framework.fuzzing_primitives.ModelWalker method*), 268
__iter__() (*framework.knowledge.feedback_collector.FeedbackCollector method*), 291
__iter__() (*framework.node.DJobGroup method*), 163
__iter_csts() (*framework.node.NodeInternals_NonTerm method*), 181
__iter_csts_verbose() (*framework.node.NodeInternals_NonTerm method*), 181
__le__() (*framework.knowledge.feedback_handler.SimilarityMetric method*), 293
__len__() (*framework.evolutionary_helpers.Population method*), 291
__lt__() (*framework.knowledge.feedback_handler.SimilarityMetric method*), 293
__lt__() (*framework.node.Node method*), 168
__module__ (*framework.comm_backends.Backend attribute*), 257
__module__ (*framework.comm_backends.BackendError attribute*), 258
__module__ (*framework.comm_backends.SSH_Backend attribute*), 259
__module__ (*framework.comm_backends.Serial_Backend attribute*), 260
__module__ (*framework.comm_backends.Shell_Backend attribute*), 261
__module__ (*framework.constraint_helpers.CSP attribute*), 296
__module__ (*framework.constraint_helpers.Constraint attribute*), 297
__module__ (*framework.constraint_helpers.ConstraintError attribute*), 298
__module__ (*framework.data.AttrGroup attribute*), 155
__module__ (*framework.data.CallBackOps attribute*), 156
__module__ (*framework.data.Data attribute*), 156
__module__ (*framework.data.DataAttr attribute*), 158
__module__ (*framework.data.DataBackend attribute*), 158
__module__ (*framework.data.DataProcess attribute*), 159
__module__ (*framework.data.EmptyBackend attribute*), 159
__module__ (*framework.data.EmptyDataProcess attribute*), 159
__module__ (*framework.data.NodeBackend attribute*), 159
__module__ (*framework.data.RawBackend attribute*), 160
__module__ (*framework.data_model.DataModel attribute*), 160
__module__ (*framework.data_model.NodeBackend attribute*), 163
__module__ (*framework.database.Database attribute*), 274
__module__ (*framework.database.FeedbackGate attribute*), 275
__module__ (*framework.dmhelpers.generic.MH attribute*), 286
__module__ (*framework.dmhelpers.generic.MH.Attr attribute*), 284
__module__ (*framework.dmhelpers.generic.MH.Charset attribute*), 285
__module__ (*framework.dmhelpers.generic.MH.Custo attribute*), 285
__module__ (*framework.dmhelpers.generic.MH.Custo.Func attribute*), 285
__module__ (*framework.dmhelpers.generic.MH.Custo.Gen attribute*), 285
__module__ (*framework.dmhelpers.generic.MH.Custo.NTerm attribute*), 285
__module__ (*framework.dmhelpers.xml.TAG_TYPE attribute*), 288
__module__ (*framework.encoders.BitReverse_Enc attribute*), 271
__module__ (*framework.encoders.Encoder attribute*), 271
__module__ (*framework.encoders.GSM7bitPacking_Enc attribute*), 272
__module__ (*framework.encoders.GSMPhoneNum_Enc attribute*), 272
__module__ (*framework.encoders.GZIP_Enc attribute*), 272
__module__ (*framework.encoders.Wrap_Enc attribute*), 273
__module__ (*framework.evolutionary_helpers.CrossoverHelper attribute*), 289
__module__ (*framework.evolutionary_helpers.CrossoverHelper.Operand attribute*), 289
__module__ (*framework.evolutionary_helpers.DefaultIndividual attribute*), 289
__module__ (*framework.evolutionary_helpers.DefaultPopulation attribute*), 289
__module__ (*framework.evolutionary_helpers.EvolutionaryScenariosFacto attribute*), 290
__module__ (*framework.evolutionary_helpers.Individual attribute*), 290
__module__ (*framework.evolutionary_helpers.Population attribute*), 291
__module__ (*framework.fuzzing_primitives.AltConfConsumer attribute*), 266
__module__ (*framework.fuzzing_primitives.BasicVisitor attribute*), 267

__module__ (framework.fuzzing_primitives.ModelWalker attribute), 268
__module__ (framework.fuzzing_primitives.NodeConsumerStub attribute), 268
__module__ (framework.fuzzing_primitives.NonTermVisitor attribute), 269
__module__ (framework.fuzzing_primitives.SeparatorDisruption attribute), 270
__module__ (framework.fuzzing_primitives.TypedNodeDisruption attribute), 270
__module__ (framework.generic_data_makers.d_add_data attribute), 219
__module__ (framework.generic_data_makers.d_call_external_program attribute), 219
__module__ (framework.generic_data_makers.d_call_function attribute), 220
__module__ (framework.generic_data_makers.d_corrupt_bits attribute), 220
__module__ (framework.generic_data_makers.d_corrupt_node_bits attribute), 220
__module__ (framework.generic_data_makers.d_fix_constraints attribute), 221
__module__ (framework.generic_data_makers.d_fuzz_model_walker attribute), 221
__module__ (framework.generic_data_makers.d_max_size attribute), 221
__module__ (framework.generic_data_makers.d_modify_nodes attribute), 222
__module__ (framework.generic_data_makers.d_next_node_model attribute), 222
__module__ (framework.generic_data_makers.d_operate_on_nodes attribute), 223
__module__ (framework.generic_data_makers.d_shallow_copy attribute), 223
__module__ (framework.generic_data_makers.d_switch_to_alternate_conf attribute), 223
__module__ (framework.generic_data_makers.g_generic_pattern attribute), 224
__module__ (framework.generic_data_makers.g_population attribute), 224
__module__ (framework.generic_data_makers.sd_constraint_fuzz attribute), 224
__module__ (framework.generic_data_makers.sd_fuzz_separator_nodes attribute), 225
__module__ (framework.generic_data_makers.sd_fuzz_typed_nodes attribute), 226
__module__ (framework.generic_data_makers.sd_struct_constraints attribute), 227
__module__ (framework.generic_data_makers.sd_switch_to_alternate_conf attribute), 227
__module__ (framework.generic_data_makers.sd_walk_csp_solutions attribute), 228
__module__ (framework.generic_data_makers.sd_walk_data_model attribute), 228
__module__ (framework.knowledge.feedback_collector.FeedbackCollector attribute), 291
__module__ (framework.knowledge.feedback_collector.FeedbackSource attribute), 292
__module__ (framework.knowledge.feedback_handler.FeedbackHandler attribute), 292
__module__ (framework.knowledge.feedback_handler.SimilarityMeasure attribute), 293
__module__ (framework.knowledge.feedback_handler.TestFbkHandler attribute), 294
__module__ (framework.knowledge.information.Hardware attribute), 294
__module__ (framework.knowledge.information.Info attribute), 294
__module__ (framework.knowledge.information.InformationCollector attribute), 295
__module__ (framework.knowledge.information.InputHandling attribute), 295
__module__ (framework.knowledge.information.Language attribute), 295
__module__ (framework.knowledge.information.OS attribute), 295
__module__ (framework.knowledge.information.OperationMode attribute), 296
__module__ (framework.knowledge.information.Test attribute), 296
__module__ (framework.knowledge.information.TrustLevel attribute), 296
__module__ (framework.logger.Logger attribute), 247
__module__ (framework.monitor.AddExistingProbeToMonitorError attribute), 249
__module__ (framework.monitor.BlockingProbeUser attribute), 249
__module__ (framework.monitor.Monitor attribute), 250
__module__ (framework.monitor.Probe attribute), 251
__module__ (framework.monitor.ProbeCmd attribute), 253
__module__ (framework.monitor.ProbeMem attribute), 254
__module__ (framework.monitor.ProbePID attribute), 255
__module__ (framework.monitor.ProbeStatus attribute), 256
__module__ (framework.monitor.ProbeTimeoutError attribute), 256
__module__ (framework.monitor.ProbeUser attribute), 256
__module__ (framework.node.BitFieldCondition attribute), 163
__module__ (framework.node.DJobGroup attribute), 163
__module__ (framework.node.DynNode_Helpers attribute), 164
__module__ (framework.node.Env attribute), 164

__module__ (*framework.node.Env4NT attribute*), 165
__module__ (*framework.node.FuncCusto attribute*), 165
__module__ (*framework.node.GenFuncCusto attribute*), 166
__module__ (*framework.node.IntCondition attribute*), 166
__module__ (*framework.node.Node attribute*), 168
__module__ (*framework.node.NodeAbstraction attribute*), 175
__module__ (*framework.node.NodeCondition attribute*), 175
__module__ (*framework.node.NodeCustomization attribute*), 176
__module__ (*framework.node.NodeInternals attribute*), 176
__module__ (*framework.node.NodeInternalsCriteria attribute*), 178
__module__ (*framework.node.NodeInternals_Empty attribute*), 178
__module__ (*framework.node.NodeInternals_Func attribute*), 179
__module__ (*framework.node.NodeInternals_GenFunc attribute*), 179
__module__ (*framework.node.NodeInternals_NonTerm attribute*), 181
__module__ (*framework.node.NodeInternals_NonTerm.NodeArgs attribute*), 181
__module__ (*framework.node.NodeInternals_Term attribute*), 184
__module__ (*framework.node.NodeInternals_TypedValue attribute*), 185
__module__ (*framework.node.NodeSemantics attribute*), 186
__module__ (*framework.node.NodeSemanticsCriteria attribute*), 187
__module__ (*framework.node.NodeSeparator attribute*), 187
__module__ (*framework.node.NonTermCusto attribute*), 188
__module__ (*framework.node.RawCondition attribute*), 188
__module__ (*framework.node.SyncExistenceObj attribute*), 189
__module__ (*framework.node.SyncObj attribute*), 189
__module__ (*framework.node.SyncQtyFromObj attribute*), 189
__module__ (*framework.node.SyncScope attribute*), 190
__module__ (*framework.node.SyncSizeObj attribute*), 190
__module__ (*framework.node_builder.NodeBuilder attribute*), 191
__module__ (*framework.node_builder.RegexParser attribute*), 198
__module__ (*framework.node_builder.RegexParser.Brackets attribute*), 193
__module__ (*framework.node_builder.RegexParser.Brackets.Comma attribute*), 192
__module__ (*framework.node_builder.RegexParser.Brackets.Final attribute*), 192
__module__ (*framework.node_builder.RegexParser.Brackets.Initial attribute*), 192
__module__ (*framework.node_builder.RegexParser.Brackets.Max attribute*), 193
__module__ (*framework.node_builder.RegexParser.Brackets.Min attribute*), 193
__module__ (*framework.node_builder.RegexParser.Choice attribute*), 193
__module__ (*framework.node_builder.RegexParser.Dot attribute*), 194
__module__ (*framework.node_builder.RegexParser.Escape attribute*), 194
__module__ (*framework.node_builder.RegexParser.EscapeMetaSequence attribute*), 194
__module__ (*framework.node_builder.RegexParser.Final attribute*), 194
__module__ (*framework.node_builder.RegexParser.Group attribute*), 194
__module__ (*framework.node_builder.RegexParser.Initial attribute*), 194
__module__ (*framework.node_builder.RegexParser.Main attribute*), 195
__module__ (*framework.node_builder.RegexParser.Parenthesis attribute*), 196
__module__ (*framework.node_builder.RegexParser.Parenthesis.Choice attribute*), 195
__module__ (*framework.node_builder.RegexParser.Parenthesis.Escape attribute*), 195
__module__ (*framework.node_builder.RegexParser.Parenthesis.Final attribute*), 196
__module__ (*framework.node_builder.RegexParser.Parenthesis.Initial attribute*), 196
__module__ (*framework.node_builder.RegexParser.Parenthesis.Main attribute*), 196
__module__ (*framework.node_builder.RegexParser.QtyState attribute*), 196
__module__ (*framework.node_builder.RegexParser.SquareBrackets attribute*), 198
__module__ (*framework.node_builder.RegexParser.SquareBrackets.AfterR attribute*), 197
__module__ (*framework.node_builder.RegexParser.SquareBrackets.Before attribute*), 197
__module__ (*framework.node_builder.RegexParser.SquareBrackets.Escape attribute*), 197
__module__ (*framework.node_builder.RegexParser.SquareBrackets.Escape attribute*), 197
__module__ (*framework.node_builder.RegexParser.SquareBrackets.Escape attribute*), 198
__module__ (*framework.node_builder.RegexParser.SquareBrackets.Final*

attribute), 198
__module__ (*framework.node_builder.RegexParser.SquareBrackets attribute*), 198
__module__ (*framework.node_builder.RegexParser.SquareBrackets attribute*), 198
__module__ (*framework.node_builder.State attribute*), 199
__module__ (*framework.node_builder.StateMachine attribute*), 199
__module__ (*framework.operator_helpers.LastInstruction attribute*), 245
__module__ (*framework.operator_helpers.Operation attribute*), 246
__module__ (*framework.operator_helpers.Operator attribute*), 246
__module__ (*framework.project.Project attribute*), 244
__module__ (*framework.scenario.FinalStep attribute*), 276
__module__ (*framework.scenario.NoDataStep attribute*), 277
__module__ (*framework.scenario.Periodic attribute*), 277
__module__ (*framework.scenario.Scenario attribute*), 278
__module__ (*framework.scenario.ScenarioEnv attribute*), 279
__module__ (*framework.scenario.Step attribute*), 280
__module__ (*framework.scenario.StepStub attribute*), 282
__module__ (*framework.scenario.Transition attribute*), 282
__module__ (*framework.tactics_helpers.DataMaker attribute*), 261
__module__ (*framework.tactics_helpers.DataMakerAttr attribute*), 262
__module__ (*framework.tactics_helpers.Disruptor attribute*), 262
__module__ (*framework.tactics_helpers.DynGenerator attribute*), 262
__module__ (*framework.tactics_helpers.DynGeneratorFromScenario attribute*), 263
__module__ (*framework.tactics_helpers.Generator attribute*), 264
__module__ (*framework.tactics_helpers.StatefulDisruptor attribute*), 264
__module__ (*framework.tactics_helpers.Tactics attribute*), 265
__module__ (*framework.tactics_helpers.dyn_generator attribute*), 266
__module__ (*framework.tactics_helpers.dyn_generator_from_scenario attribute*), 266
__module__ (*framework.target_helpers.EmptyTarget attribute*), 229
__module__ (*framework.target_helpers.Target attribute*), 230
__module__ (*framework.target_helpers.TargetError attribute*), 233
__module__ (*framework.target_helpers.TargetNotReady attribute*), 233
__module__ (*framework.target_helpers.TargetStuck attribute*), 233
__module__ (*framework.targets.debug.IncorrectTargetError attribute*), 242
__module__ (*framework.targets.debug.ShmemMappingError attribute*), 242
__module__ (*framework.targets.debug.TestTarget attribute*), 242
__module__ (*framework.targets.local.LocalTarget attribute*), 238
__module__ (*framework.targets.network.NetworkTarget attribute*), 234
__module__ (*framework.targets.printer.PrinterTarget attribute*), 241
__module__ (*framework.targets.ssh.SSHTarget attribute*), 240
__module__ (*framework.value_types.BitField attribute*), 200
__module__ (*framework.value_types.Filename attribute*), 202
__module__ (*framework.value_types.FolderPath attribute*), 204
__module__ (*framework.value_types.GSM7bitPacking attribute*), 205
__module__ (*framework.value_types.GSMPhoneNum attribute*), 205
__module__ (*framework.value_types.GZIP attribute*), 205
__module__ (*framework.value_types.INT attribute*), 206
__module__ (*framework.value_types.INT16 attribute*), 207
__module__ (*framework.value_types.INT32 attribute*), 207
__module__ (*framework.value_types.INT64 attribute*), 207
__module__ (*framework.value_types.INT8 attribute*), 208
__module__ (*framework.value_types.INT_str attribute*), 208
__module__ (*framework.value_types.SINT16_be attribute*), 209
__module__ (*framework.value_types.SINT16_le attribute*), 209
__module__ (*framework.value_types.SINT32_be attribute*), 209
__module__ (*framework.value_types.SINT32_le attribute*), 210
__module__ (*framework.value_types.SINT64_be attribute*), 210

__module__ (*framework.value_types.SINT64_le attribute*), 210
__module__ (*framework.value_types.SINT8 attribute*), 210
__module__ (*framework.value_types.String attribute*), 212
__module__ (*framework.value_types.UINT16_be attribute*), 215
__module__ (*framework.value_types.UINT16_le attribute*), 215
__module__ (*framework.value_types.UINT32_be attribute*), 216
__module__ (*framework.value_types.UINT32_le attribute*), 216
__module__ (*framework.value_types.UINT64_be attribute*), 216
__module__ (*framework.value_types.UINT64_le attribute*), 217
__module__ (*framework.value_types.UINT8 attribute*), 217
__module__ (*framework.value_types.VT attribute*), 217
__module__ (*framework.value_types.VT_Alt attribute*), 218
__module__ (*framework.value_types.Wrapper attribute*), 219
__new__() (*framework.tactics_helpers.dyn_generator_from_scenario static method*), 266
__next__() (*framework.evolutionary_helpers.Population method*), 291
__post_handling() (*framework.node_builder.NodeBuilder method*), 191
__pre_handling() (*framework.node_builder.NodeBuilder method*), 191
__register_new_data_maker() (*framework.tactics_helpers.Tactics method*), 265
__repr__() (*framework.data.Data method*), 156
__repr__() (*framework.data.DataProcess method*), 159
__repr__() (*framework.evolutionary_helpers.DefaultPopulation method*), 289
__repr__() (*framework.evolutionary_helpers.Population method*), 291
__repr__() (*framework.node.DJobGroup method*), 163
__repr__() (*framework.value_types.String method*), 212
__reversed__() (*framework.node.DJobGroup method*), 163
__set_current_internals() (*framework.node.Node method*), 168
__set_data_maker_weight() (*framework.tactics_helpers.Tactics method*), 265
__setitem__() (*framework.evolutionary_helpers.Population method*), 291
__setitem__() (*framework.node.Node method*), 168
__str__() (*framework.data.Data method*), 156
__str__() (*framework.data_model.DataModel method*), 160
__str__() (*framework.knowledge.feedback_collector.FeedbackSource method*), 292
__str__() (*framework.knowledge.information.InformationCollector method*), 295
__str__() (*framework.logger.Logger method*), 247
__str__() (*framework.monitor.Probe method*), 251
__str__() (*framework.node.Node method*), 168
__str__() (*framework.node.NodeSemantics method*), 186
__str__() (*framework.operator_helpers.Operator method*), 246
__str__() (*framework.scenario.Periodic method*), 277
__str__() (*framework.scenario.Scenario method*), 278
__str__() (*framework.scenario.Step method*), 280
__str__() (*framework.scenario.Transition method*), 282
__str__() (*framework.target_helpers.Target method*), 230
_add_default_crossover_info() (*framework.evolutionary_helpers.CrossoverHelper static method*), 289
_add_info() (*framework.generic_data_makers.d_modify_nodes method*), 222
_add_info() (*framework.generic_data_makers.d_operate_on_nodes method*), 223
_add_separator_cases() (*framework.fuzzing_primitives.TypedNodeDisruption method*), 270
_alter_data_step() (*framework.tactics_helpers.DynGeneratorFromScenario method*), 263
_alter_transition_conditions() (*framework.tactics_helpers.DynGeneratorFromScenario method*), 263
_altered_data_queued (*framework.target_helpers.Target attribute*), 230
_args_desc (*framework.generic_data_makers.d_add_data attribute*), 219
_args_desc (*framework.generic_data_makers.d_call_external_program attribute*), 219
_args_desc (*framework.generic_data_makers.d_call_function attribute*), 220
_args_desc (*framework.generic_data_makers.d_corrupt_bits_by_position attribute*), 220
_args_desc (*framework.generic_data_makers.d_corrupt_node_bits attribute*), 220
_args_desc (*framework.generic_data_makers.d_fix_constraints attribute*), 221
_args_desc (*framework.generic_data_makers.d_fuzz_model_structure attribute*), 221

_args_desc (*framework.generic_data_makers.d_max_size attribute*), 221
_args_desc (*framework.generic_data_makers.d_modify_nodes attribute*), 222
_args_desc (*framework.generic_data_makers.d_next_node_content attribute*), 222
_args_desc (*framework.generic_data_makers.d_operate_on_nodes attribute*), 223
_args_desc (*framework.generic_data_makers.d_shallow_copy attribute*), 223
_args_desc (*framework.generic_data_makers.d_switch_to_alternate_conf attribute*), 223
_args_desc (*framework.generic_data_makers.g_generic_pattern attribute*), 224
_args_desc (*framework.generic_data_makers.g_population attribute*), 224
_args_desc (*framework.generic_data_makers.sd_constraint_fuzz attribute*), 224
_args_desc (*framework.generic_data_makers.sd_fuzz_separator_node attribute*), 225
_args_desc (*framework.generic_data_makers.sd_fuzz_typed_nodes attribute*), 226
_args_desc (*framework.generic_data_makers.sd_struct_constraints attribute*), 227
_args_desc (*framework.generic_data_makers.sd_switch_to_clone_ctxt attribute*), 227
_args_desc (*framework.generic_data_makers.sd_walk_csp_solutions attribute*), 228
_args_desc (*framework.generic_data_makers.sd_walk_data_model attribute*), 228
_args_desc (*framework.operator_helpers.Operator attribute*), 246
_args_desc (*framework.tactics_helpers.DataMaker attribute*), 261
_args_desc (*framework.tactics_helpers.DynGenerator attribute*), 262
_args_desc (*framework.tactics_helpers.DynGeneratorFromScenario attribute*), 263
_atom_absorption_additional_actions() (*framework.data_model.DataModel method*), 160
_backend() (*framework.data_model.DataModel method*), 160
_before_sending_data() (*framework.targets.local.LocalTarget method*), 238
_before_sending_data() (*framework.targets.network.NetworkTarget method*), 234
_bytes2str() (*framework.value_types.String method*), 212
_callback_dispatcher_after_fbk() (*framework.tactics_helpers.DynGeneratorFromScenario method*), 263
_callback_dispatcher_after_sending() (*framework.tactics_helpers.DynGeneratorFromScenario method*), 263
_callback_dispatcher_before_sending_step1() (*framework.tactics_helpers.DynGeneratorFromScenario method*), 263
_callback_dispatcher_before_sending_step2() (*framework.tactics_helpers.DynGeneratorFromScenario method*), 263
_callback_dispatcher_final() (*framework.tactics_helpers.DynGeneratorFromScenario method*), 263
_check_alphabet() (*framework.value_types.String method*), 212
_check_conf() (*framework.node.Node method*), 169
_check_constraints() (*framework.value_types.BitField method*), 200
_check_constraints() (*framework.value_types.String method*), 212
_check_constraints_and_update() (*framework.value_types.INT method*), 206
_check_constraints_and_update() (*framework.value_types.String method*), 212
_check_contents() (*framework.value_types.String method*), 212
_check_data_for_fuzz_completion_cbk() (*framework.tactics_helpers.DynGeneratorFromScenario method*), 263
_check_inclusion() (*framework.node.NodeCondition method*), 175
_check_int() (*framework.node.NodeCondition method*), 175
_check_size_constraints() (*framework.value_types.String method*), 212
_cleanup() (*framework.tactics_helpers.Disruptor method*), 262
_cleanup() (*framework.tactics_helpers.Generator method*), 264
_cleanup() (*framework.tactics_helpers.StatefulDisruptor method*), 264
_cleanup_delayed_nodes() (*framework.node.NodeInternals_NonTerm static method*), 181
_cleanup_entangled_nodes() (*framework.node.NodeInternals_NonTerm method*), 181
_cleanup_entangled_nodes_from() (*framework.node.NodeInternals_NonTerm method*), 181
_cleanup_state() (*framework.targets.network.NetworkTarget method*), 234
_cleanup_walking_attrs() (*framework.tactics_helpers.DynGeneratorFromScenario method*), 263

_clear() (*framework.monitor.BlockingProbeUser method*), 249
_clear() (*framework.monitor.ProbeUser method*), 256
_clear_attr_direct() (*framework.node.NodeInternals method*), 176
_clear_drawn_node_attrs() (*framework.node.NodeInternals_NonTerm method*), 181
_clone_from_dict() (*framework.node_builder.NodeBuilder method*), 191
_clone_node() (*framework.node.NodeInternals_NonTerm method*), 181
_clone_node_cleanup() (*framework.node.NodeInternals_NonTerm method*), 181
_clone_separator() (*framework.node.NodeInternals_NonTerm method*), 181
_clone_separator_cleanup() (*framework.node.NodeInternals_NonTerm method*), 181
_collect_fbk_loop() (*framework.targets.debug.TestTarget method*), 242
_collect_feedback_from() (*framework.targets.network.NetworkTarget method*), 235
_complete_func() (*framework.node_builder.NodeBuilder method*), 191
_complete_generator() (*framework.node_builder.NodeBuilder method*), 191
_complete_generator_from_desc() (*framework.node_builder.NodeBuilder method*), 191
_compute_confs() (*framework.node.Node method*), 169
_compute_probability_of_survival() (*framework.evolutionary_helpers.DefaultPopulation method*), 289
_compute_scores() (*framework.evolutionary_helpers.DefaultPopulation method*), 289
_condition_satisfied() (*framework.node.SyncExistenceObj method*), 189
_connect_to_additional_feedback_sockets() (*framework.targets.network.NetworkTarget method*), 235
_connect_to_target() (*framework.targets.network.NetworkTarget method*), 235
_constraints (*framework.constraint_helpers.CSP attribute*), 296
_construct_subnodes() (*framework.node.NodeInternals_NonTerm method*), 181
_convert_to_internal_repr() (*framework.node.NodeInternals_Term static method*), 184
_convert_value() (*framework.value_types.INT method*), 206
_convert_value() (*framework.value_types.INT_str method*), 208
_copy_nodelist() (*framework.node.NodeInternals_NonTerm method*), 181
_count_brothers() (*framework.evolutionary_helpers.CrossoverHelper.Operand method*), 289
_create_atom_from_raw_data_specific() (*framework.data_model.DataModel method*), 161
_create_generator_node() (*framework.node_builder.NodeBuilder method*), 191
_create_graph_from_desc() (*framework.node_builder.NodeBuilder method*), 191
_create_leaf_node() (*framework.node_builder.NodeBuilder method*), 191
_create_nodes_from_shape() (*framework.node_builder.NodeBuilder method*), 191
_create_non_terminal_node() (*framework.node_builder.NodeBuilder method*), 191
_create_non_terminal_node() (*framework.node_builder.RegexParser method*), 199
_create_non_terminal_node_from_regex() (*framework.node_builder.NodeBuilder method*), 191
_create_pkey() (*framework.comm_backends.SSH_Backend static method*), 259
_create_terminal_node() (*framework.node_builder.RegexParser method*), 199
_create_todo_list() (*framework.node_builder.NodeBuilder method*), 191
_crossover() (*framework.evolutionary_helpers.DefaultPopulation method*), 290
_crossover_algo2() (*framework.evolutionary_helpers.CrossoverHelper*

class method), 289
_current_qty (*framework.node.NodeInternals_NonTerm.NodeAttrs attribute*), 181
_custo_items (*framework.node.FuncCusto attribute*), 165
_custo_items (*framework.node.GenFuncCusto attribute*), 166
_custo_items (*framework.node.NodeCustomization attribute*), 176
_custo_items (*framework.node.NonTermCusto attribute*), 188
_custom_data_handling_before_emission() (*framework.targets.network.NetworkTarget method*), 235
_default_qty (*framework.node.NodeInternals_NonTerm.NodeAttrs attribute*), 181
_delay_sending() (*framework.plumbing.FmkPlumbing method*), 298
_do_reset() (*framework.fuzzing_primitives.ModelWalker method*), 268
_empty_data_backend (*framework.data.Data attribute*), 156
_enable_fuzz_mode() (*framework.value_types.BitField method*), 200
_enable_fuzz_mode() (*framework.value_types.String method*), 212
_enable_fuzz_mode() (*framework.value_types.VT_Alt method*), 218
_enable_normal_mode() (*framework.value_types.BitField method*), 200
_enable_normal_mode() (*framework.value_types.String method*), 212
_enable_normal_mode() (*framework.value_types.VT_Alt method*), 218
_encode_bitfield() (*framework.value_types.BitField method*), 200
_encode_target_feedback() (*framework.logger.Logger method*), 247
_encoder_arg (*framework.value_types.GSM7bitPacking attribute*), 205
_encoder_arg (*framework.value_types.GSMPhoneNum attribute*), 205
_encoder_arg (*framework.value_types.GZIP attribute*), 205
_encoder_arg (*framework.value_types.Wrapper attribute*), 219
_encoder_cls (*framework.value_types.GSM7bitPacking attribute*), 205
_encoder_cls (*framework.value_types.GSMPhoneNum attribute*), 205
_encoder_cls (*framework.value_types.GZIP attribute*), 205
_encoder_cls (*framework.value_types.String attribute*), 212
_encoder_cls (*framework.value_types.Wrapper attribute*), 219
_encoder_obj (*framework.value_types.String attribute*), 212
_ensure_enc_sizes_consistency() (*framework.value_types.String method*), 212
_exec_command() (*framework.comm_backends.Backend method*), 257
_exec_command() (*framework.comm_backends.SSH_Backend method*), 259
_exec_command() (*framework.comm_backends.Serial_Backend method*), 260
_exec_command() (*framework.comm_backends.Shell_Backend method*), 261
_exhausted_solutions (*framework.constraint_helpers.CSP attribute*), 296
_existence_from_node() (*framework.node.NodeInternals_NonTerm static method*), 182
_expand_delayed_nodes() (*framework.node.NodeInternals_NonTerm static method*), 182
_export_data_func() (*framework.logger.Logger method*), 247
_extensions (*framework.target_helpers.Target attribute*), 230
_feedback_collect() (*framework.targets.network.NetworkTarget method*), 235
_feedback_complete() (*framework.targets.network.NetworkTarget method*), 235
_feedback_handling() (*framework.targets.network.NetworkTarget method*), 235
_feedback_mode (*framework.target_helpers.EmptyTarget attribute*), 229
_feedback_mode (*framework.target_helpers.Target attribute*), 230
_feedback_mode (*framework.targets.debug.TestTarget attribute*), 242
_feedback_mode (*framework.targets.local.LocalTarget attribute*), 238
_feedback_mode (*frame-*

work.targets.network.NetworkTarget attribute), 235
_feedback_mode (*framework.targets.printer.PrinterTarget attribute*), 241
_feedback_processing() (*framework.project.Project method*), 244
_finalize_nonterm_node() (*framework.node.Node method*), 169
_forward_data() (*framework.targets.debug.TestTarget method*), 242
_get_additional_feedback_sockets() (*framework.targets.network.NetworkTarget method*), 235
_get_all_paths_rec() (*framework.node.Node method*), 169
_get_cmd() (*framework.generic_data_makers.d_call_external_program method*), 219
_get_color_function() (*framework.database.Database method*), 274
_get_data_semantic_key() (*framework.targets.network.NetworkTarget method*), 235
_get_delayed_value() (*framework.node.NodeInternals_GenFunc method*), 179
_get_from_dict() (*framework.node_builder.NodeBuilder method*), 191
_get_graph_info() (*framework.node.DynNode_Helpers method*), 164
_get_heavier_component() (*framework.node.NodeInternals_NonTerm method*), 182
_get_info_from_subnode_description() (*framework.node.NodeInternals_NonTerm method*), 182
_get_mem() (*framework.monitor.ProbeMem method*), 254
_get_net_info_from() (*framework.targets.network.NetworkTarget method*), 235
_get_next_heavier_component() (*framework.node.NodeInternals_NonTerm static method*), 182
_get_next_random_component() (*framework.node.NodeInternals_NonTerm static method*), 182
_get_node_and_minmax_from() (*framework.node.NodeInternals_NonTerm method*), 182
_get_node_from() (*framework.node.NodeInternals_NonTerm method*), 182
_get_nodes() (*framework.evolutionary_helpers.CrossoverHelper static method*), 289
_get_path_depth() (*framework.value_types.Filename method*), 202
_get_path_from_value() (*framework.value_types.Filename method*), 202
_get_pid() (*framework.monitor.ProbePID method*), 255
_get_probe_ref() (*framework.monitor.Monitor method*), 250
_get_random_component() (*framework.node.NodeInternals_NonTerm method*), 182
_get_socket_type() (*framework.targets.network.NetworkTarget method*), 235
_get_value() (*framework.node.Node method*), 169
_get_value() (*framework.node.NodeInternals method*), 176
_get_value() (*framework.node.NodeInternals_Empty method*), 178
_get_value() (*framework.node.NodeInternals_GenFunc method*), 179
_get_value() (*framework.node.NodeInternals_NonTerm method*), 182
_get_value() (*framework.node.NodeInternals_Term method*), 184
_get_value_specific() (*framework.node.NodeInternals_Func method*), 179
_get_value_specific() (*framework.node.NodeInternals_Term method*), 184
_get_value_specific() (*framework.node.NodeInternals_TypedValue method*), 185
_go_on() (*framework.monitor.ProbeUser method*), 256
_graph_setup() (*framework.scenario.Scenario method*), 278
_handle_attrs() (*framework.dmhelpers.generic.MH static method*), 286
_handle_binary_content() (*framework.database.Database method*), 274
_handle_binary_content() (*framework.logger.Logger method*), 247
_handle_common_attr() (*framework.node_builder.NodeBuilder method*), 191
_handle_cond() (*framework.node.RawCondition method*), 188
_handle_connection_to_fbk_server() (*frame-*

work.targets.network.NetworkTarget method), 235
_handle_custo() (*framework.node_builder.NodeBuilder method*), 191
_handle_data_desc() (*framework.scenario.Step method*), 280
_handle_exception() (*framework.monitor.ProbeUser method*), 256
_handle_fbk() (*framework.targets.debug.TestTarget method*), 242
_handle_name() (*framework.node_builder.NodeBuilder method*), 191
_handle_target_connection() (*framework.targets.network.NetworkTarget method*), 235
_handle_user_input() (*framework.node.NodeInternalsCriteria method*), 178
_handle_user_input() (*framework.node.NodeSemanticsCriteria method*), 187
_handle_user_inputs() (*in module framework.tactics_helpers*), 266
_init_main_properties() (*framework.scenario.Scenario method*), 278
_init_reinit_seq_properties() (*framework.scenario.Scenario method*), 278
_init_specific() (*framework.node.NodeInternals method*), 176
_init_specific() (*framework.node.NodeInternals_Func method*), 179
_init_specific() (*framework.node.NodeInternals_GenFunc method*), 179
_init_specific() (*framework.node.NodeInternals_NonTerm method*), 182
_init_specific() (*framework.node.NodeInternals_Term method*), 184
_init_specific() (*framework.node.NodeInternals_TypedValue method*), 186
_initialize() (*framework.evolutionary_helpers.DefaultPopulation method*), 290
_initialize() (*framework.evolutionary_helpers.Population method*), 291
_is_solution_queried (*framework.constraint_helpers.CSP attribute*), 296
_is_valid() (*framework.database.Database method*), 274
_is_valid_socket_type() (*framework.targets.network.NetworkTarget method*), 235
_kill() (*framework.evolutionary_helpers.DefaultPopulation method*), 290
_last_ack_date (*framework.targets.debug.TestTarget attribute*), 242
_last_sending_date (*framework.target_helpers.Target attribute*), 230
_listen_to_target() (*framework.targets.network.NetworkTarget method*), 235
_log_directly_retrieved_target_feedback() (*framework.plumbing.FmkPlumbing method*), 298
_log_feedback() (*framework.logger.Logger method*), 247
_log_handler() (*framework.logger.Logger method*), 247
_logger (*framework.target_helpers.Target attribute*), 230
_make_private_specific() (*framework.node.NodeInternals method*), 176
_make_private_specific() (*framework.node.NodeInternals_GenFunc method*), 179
_make_private_specific() (*framework.node.NodeInternals_NonTerm method*), 182
_make_private_specific() (*framework.node.NodeInternals_Term method*), 184
_make_private_term_specific() (*framework.node.NodeInternals_Func method*), 179
_make_private_term_specific() (*framework.node.NodeInternals_Term method*), 185
_make_private_term_specific() (*framework.node.NodeInternals_TypedValue method*), 186
_make_specific() (*framework.node.NodeInternals method*), 176
_make_specific() (*framework.node.NodeInternals_GenFunc method*), 179
_make_specific() (*framework.node.NodeInternals_NonTerm method*), 182
_make_specific() (*framework.node.NodeInternals_TypedValue method*), 186

_make_step_stutter() (*framework.tactics_helpers.DynGeneratorFromScenario method*), 263
_map_input_shmem() (*framework.targets.debug.TestTarget method*), 242
_match_exclusive_criteria() (*framework.node.NodeSemantics method*), 186
_match_mandatory_attrs() (*framework.node.NodeInternals method*), 176
_match_mandatory_criteria() (*framework.node.NodeSemantics method*), 186
_match_mandatory_custo() (*framework.node.NodeInternals method*), 176
_match_negative_attrs() (*framework.node.NodeInternals method*), 176
_match_negative_criteria() (*framework.node.NodeSemantics method*), 186
_match_negative_custo() (*framework.node.NodeInternals method*), 176
_match_negative_node_kinds() (*framework.node.NodeInternals method*), 177
_match_negative_node_subkinds() (*framework.node.NodeInternals method*), 177
_match_node_constraints() (*framework.node.NodeInternals method*), 177
_match_node_kinds() (*framework.node.NodeInternals method*), 177
_match_node_subkinds() (*framework.node.NodeInternals method*), 177
_match_optionalbut1_criteria() (*framework.node.NodeSemantics method*), 186
_max (*framework.node.NodeInternals_NonTerm.NodeAttrs attribute*), 181
_merge_brothers() (*framework.evolutionary_helpers.CrossoverHelper.Operand method*), 289
_min (*framework.node.NodeInternals_NonTerm.NodeAttrs attribute*), 181
_model (*framework.constraint_helpers.CSP attribute*), 296
_modelwalker_user (*framework.generic_data_makers.d_add_data attribute*), 219
_modelwalker_user (*framework.generic_data_makers.d_call_external_program attribute*), 219
_modelwalker_user (*framework.generic_data_makers.d_call_function attribute*), 220
_modelwalker_user (*framework.generic_data_makers.d_corrupt_bits_by_position attribute*), 220
_modelwalker_user (*framework.generic_data_makers.d_corrupt_node_bits attribute*), 220
_modelwalker_user (*framework.generic_data_makers.d_fix_constraints attribute*), 221
_modelwalker_user (*framework.generic_data_makers.d_fuzz_model_structure attribute*), 221
_modelwalker_user (*framework.generic_data_makers.d_max_size attribute*), 221
_modelwalker_user (*framework.generic_data_makers.d_modify_nodes attribute*), 222
_modelwalker_user (*framework.generic_data_makers.d_next_node_content attribute*), 222
_modelwalker_user (*framework.generic_data_makers.d_operate_on_nodes attribute*), 223
_modelwalker_user (*framework.generic_data_makers.d_shallow_copy attribute*), 223
_modelwalker_user (*framework.generic_data_makers.d_switch_to_alternate_conf attribute*), 223
_modelwalker_user (*framework.generic_data_makers.g_generic_pattern attribute*), 224
_modelwalker_user (*framework.generic_data_makers.g_population attribute*), 224
_modelwalker_user (*framework.generic_data_makers.sd_constraint_fuzz attribute*), 225
_modelwalker_user (*framework.generic_data_makers.sd_fuzz_separator_nodes attribute*), 225
_modelwalker_user (*framework.generic_data_makers.sd_fuzz_typed_nodes attribute*), 226
_modelwalker_user (*framework.generic_data_makers.sd_struct_constraints attribute*), 227
_modelwalker_user (*framework.generic_data_makers.sd_switch_to_alternate_conf attribute*), 227
_modelwalker_user (*framework.generic_data_makers.sd_walk_csp_solutions attribute*), 228
_modelwalker_user (*framework.generic_data_makers.sd_walk_data_model attribute*), 229
_modelwalker_user (*framework.tactics_helpers.DataMaker attribute*),

261
_mutate() (framework.evolutionary_helpers.DefaultPopulation method), 290
_negated_relation() (framework.constraint_helpers.Constraint method), 297
_notify_armed() (framework.monitor.BlockingProbeUser method), 249
_notify_probe_started() (framework.monitor.ProbeUser method), 257
_notify_status_retrieved() (framework.monitor.BlockingProbeUser method), 249
_orig_relation (framework.constraint_helpers.Constraint attribute), 297
_orig_var_domain (framework.constraint_helpers.CSP attribute), 296
_parse_node_desc() (framework.node.NodeInternals_NonTerm method), 182
_pending_data (framework.target_helpers.Target attribute), 230
_pending_data_id (framework.target_helpers.Target attribute), 230
_planned_reset (framework.node.NodeInternals_NonTerm.NodeAttrs attribute), 181
_populate_fuzzy_vt_list() (framework.fuzzing_primitives.TypedNodeDisruption method), 270
_populate_values() (framework.value_types.String method), 212
_post_freeze() (framework.node.Node method), 169
_post_freeze_handler (framework.node.Node attribute), 167
_precondition_subnode_ops() (framework.node.NodeInternals_NonTerm method), 182
_prepare_format_str() (framework.value_types.INT_str method), 208
_previous_current_qty_was_none (framework.node.NodeInternals_NonTerm.NodeAttrs attribute), 181
_previous_qty (framework.node.NodeInternals_NonTerm.NodeAttrs attribute), 181
_print() (framework.node.Node static method), 169
_print_console() (framework.logger.Logger method), 248
_print_contents() (framework.node.Node static method), 169
_print_name() (framework.node.Node static method), 169
_print_nonterm() (framework.node.Node static method), 169
_print_raw() (framework.node.Node static method), 169
_print_type() (framework.node.Node static method), 169
_problem (framework.constraint_helpers.CSP attribute), 296
_process_next_constraint() (framework.generic_data_makers.sd_constraint_fuzz method), 225
_process_target_feedback() (framework.logger.Logger method), 248
_qty_from_node() (framework.node.NodeInternals_NonTerm static method), 182
_qty_sequence (framework.node.NodeInternals_NonTerm.NodeAttrs attribute), 181
_raw_connect_to() (framework.targets.network.NetworkTarget method), 236
_raw_listen_to() (framework.targets.network.NetworkTarget method), 236
_raw_server_main() (framework.targets.network.NetworkTarget method), 236
_read_fd() (framework.comm_backends.SSH_Backend method), 259
_read_serial() (framework.comm_backends.Serial_Backend method), 260
_read_value_from() (framework.value_types.BitField method), 200
_read_value_from() (framework.value_types.INT method), 206
_read_value_from() (framework.value_types.INT_str method), 208
_read_value_from() (framework.value_types.String method), 212
_register_last_ack_date() (framework.targets.network.NetworkTarget method), 236
_register_todo() (framework.node_builder.NodeBuilder method), 191
_reset_depth() (framework.node.Node method), 169
_reset_idx() (framework.value_types.BitField method), 200
_reset_state_info() (framework.node.NodeInternals_NonTerm method), 182

_reset_state_specific() (framework.node.NodeInternals_Func method), 179
_reset_state_specific() (framework.node.NodeInternals_Term method), 185
_reset_state_specific() (framework.node.NodeInternals_TypedValue method), 186
_restore_dmaker_internals() (in module framework.tactics_helpers), 266
_reverse_bits() (framework.encoders.BitReverse_Enc method), 271
_run() (framework.monitor.BlockingProbeUser method), 249
_run() (framework.monitor.ProbeUser method), 257
_run() (framework.node_builder.RegexParser.Brackets.Comma method), 192
_run() (framework.node_builder.RegexParser.Brackets.Final method), 192
_run() (framework.node_builder.RegexParser.Brackets.Initial method), 193
_run() (framework.node_builder.RegexParser.Brackets.Max method), 193
_run() (framework.node_builder.RegexParser.Brackets.Min method), 193
_run() (framework.node_builder.RegexParser.Choice method), 193
_run() (framework.node_builder.RegexParser.Dot method), 194
_run() (framework.node_builder.RegexParser.Escape method), 194
_run() (framework.node_builder.RegexParser.EscapeMetaSequence method), 194
_run() (framework.node_builder.RegexParser.Final method), 194
_run() (framework.node_builder.RegexParser.Initial method), 195
_run() (framework.node_builder.RegexParser.Main method), 195
_run() (framework.node_builder.RegexParser.Parenthesis.Choice method), 195
_run() (framework.node_builder.RegexParser.Parenthesis.Escape method), 195
_run() (framework.node_builder.RegexParser.Parenthesis.Final method), 196
_run() (framework.node_builder.RegexParser.Parenthesis.Initial method), 196
_run() (framework.node_builder.RegexParser.Parenthesis.Main method), 196
_run() (framework.node_builder.RegexParser.QtyState method), 196
_run() (framework.node_builder.RegexParser.SquareBrackets.AfterRange method), 197
_run() (framework.node_builder.RegexParser.SquareBrackets.BeforeRange method), 197
_run() (framework.node_builder.RegexParser.SquareBrackets.EscapeAfter method), 197
_run() (framework.node_builder.RegexParser.SquareBrackets.EscapeBefor method), 197
_run() (framework.node_builder.RegexParser.SquareBrackets.EscapeMeta method), 198
_run() (framework.node_builder.RegexParser.SquareBrackets.Final method), 198
_run() (framework.node_builder.RegexParser.SquareBrackets.Initial method), 198
_run() (framework.node_builder.RegexParser.SquareBrackets.Range method), 198
_run() (framework.node_builder.State method), 199
_run() (framework.node_builder.StateMachine method), 200
_send_data() (framework.plumbing.FmkPlumbing method), 298
_send_data() (framework.targets.network.NetworkTarget method), 236
_send_data_lock (framework.target_helpers.Target attribute), 230
_server_main() (framework.targets.network.NetworkTarget method), 236
_set_attr_direct() (framework.node.NodeInternals method), 177
_set_clone_info() (framework.node.Node method), 169
_set_default_value() (framework.node.NodeInternals_Term method), 185
_set_default_value_specific() (framework.node.NodeInternals_Term method), 185
_set_default_value_specific() (framework.node.NodeInternals_TypedValue method), 186
_set_drawn_node_attrs() (framework.node.NodeInternals_NonTerm method), 182
_set_env() (framework.node_builder.NodeBuilder method), 191
_set_feedback_timeout_specific() (framework.target_helpers.Target method), 230
_set_feedback_timeout_specific() (framework.targets.ssh.SSHTarget method), 240
_set_frozen_value() (framework.node.NodeInternals_Term method), 185
_set_seed() (framework.tactics_helpers.StatefulDisruptor

method), 264
_set_subtrees_current_conf() (*framework.node.Node method*), 169
_set_sync_node() (*framework.node_builder.NodeBuilder method*), 191
_setup() (*framework.tactics_helpers.Disruptor method*), 262
_setup() (*framework.tactics_helpers.Generator method*), 264
_setup() (*framework.tactics_helpers.StatefulDisruptor method*), 264
_setup_constraints() (*framework.node_builder.NodeBuilder method*), 191
_size_from_node() (*framework.node.NodeInternals_NonTerm static method*), 182
_solutions (*framework.constraint_helpers.CSP attribute*), 296
_solve_constraints() (*framework.constraint_helpers.CSP method*), 296
_sql_handler() (*framework.database.Database method*), 274
_start() (*framework.comm_backends.Backend method*), 257
_start() (*framework.comm_backends.SSH_Backend method*), 259
_start() (*framework.comm_backends.Serial_Backend method*), 260
_start() (*framework.comm_backends.Shell_Backend method*), 261
_start() (*framework.knowledge.feedback_handler.FeedbackHandler method*), 292
_start() (*framework.monitor.Probe method*), 251
_start() (*framework.operator_helpers.Operator method*), 246
_start() (*framework.target_helpers.Target method*), 230
_start_fbk_collector() (*framework.targets.network.NetworkTarget method*), 236
_started (*framework.target_helpers.Target attribute*), 230
_stop() (*framework.comm_backends.Backend method*), 257
_stop() (*framework.comm_backends.SSH_Backend method*), 259
_stop() (*framework.comm_backends.Serial_Backend method*), 260
_stop() (*framework.comm_backends.Shell_Backend method*), 261
_stop() (*framework.knowledge.feedback_handler.FeedbackHandler method*), 292
_stop() (*framework.monitor.Probe method*), 251
_stop() (*framework.target_helpers.Target method*), 230
_stop_log_handler() (*framework.logger.Logger method*), 248
_stop_sql_handler() (*framework.database.Database method*), 274
_str2bytes() (*framework.value_types.String method*), 212
_stutter_cbk() (*framework.scenario.Step method*), 280
_stutter_cbk() (*framework.tactics_helpers.DynGeneratorFromScenario method*), 263
_swap_nodes() (*framework.evolutionary_helpers.CrossoverHelper static method*), 289
_sync_nodes_specific() (*framework.node.SyncObj method*), 189
_sync_nodes_specific() (*framework.node.SyncSizeObj method*), 190
_tobytes() (*framework.node.Node method*), 169
_unconvert_value() (*framework.value_types.INT method*), 206
_unconvert_value() (*framework.value_types.INT_str method*), 208
_unfreeze_reevaluate_constraints() (*framework.node.NodeInternals_Func method*), 179
_unfreeze_reevaluate_constraints() (*framework.node.NodeInternals_Term method*), 185
_unfreeze_reevaluate_constraints() (*framework.node.NodeInternals_TypedValue method*), 186
_unfreeze_without_state_change() (*framework.node.NodeInternals_Func method*), 179
_unfreeze_without_state_change() (*framework.node.NodeInternals_Term method*), 185
_unfreeze_without_state_change() (*framework.node.NodeInternals_TypedValue method*), 186
_unmake_specific() (*framework.node.NodeInternals method*), 177
_unmake_specific() (*framework.node.NodeInternals_GenFunc method*), 179
_unmake_specific() (*framework.node.NodeInternals_NonTerm method*), 182
_unmake_specific() (*framework.node.NodeInternals_TypedValue method*), 186

_update_csp() (framework.generic_data_makers.sd_constraint_fuzz method), 225
_update_dyn_helper() (framework.node.DynNode_Helpers method), 164
_update_node_refs() (framework.node.NodeInternals method), 177
_update_provided_node() (framework.node_builder.NodeBuilder method), 192
_update_value_specific() (framework.node.NodeInternals_Term method), 185
_update_value_specific() (framework.node.NodeInternals_TypedValue method), 186
_user_input_conformity() (in module framework.tactics_helpers), 266
_validate_int_vt() (framework.dmhelpers.generic.MH static method), 286
_validate_vt() (framework.dmhelpers.generic.MH static method), 286
_var_domain (framework.constraint_helpers.CSP attribute), 296
_var_domain (framework.constraint_helpers.Constraint attribute), 297
_var_domain_updated (framework.constraint_helpers.CSP attribute), 296
_var_node_mapping (framework.constraint_helpers.CSP attribute), 296
_var_to_varns (framework.constraint_helpers.CSP attribute), 296
_vars (framework.constraint_helpers.CSP attribute), 297
_verify_keys_conformity() (framework.node_builder.NodeBuilder method), 192
_view_linux() (framework.scenario.Scenario method), 278
_view_windows() (framework.scenario.Scenario method), 278
_wait() (framework.monitor.ProbeUser method), 257
_wait_for_data_ready() (framework.monitor.BlockingProbeUser method), 249
_wait_for_fmk_sync() (framework.monitor.BlockingProbeUser method), 250
_wait_for_probe() (framework.monitor.ProbeUser method), 257
_wait_for_specific_probes() (framework.monitor.Monitor method), 250

A

Abs_Postpone (framework.dmhelpers.generic.MH.Attr attribute), 284
Abs_Postpone (framework.node.NodeInternals attribute), 176
absorb() (framework.data_model.DataModel method), 161
absorb() (framework.node.Node method), 169
absorb() (framework.node.NodeInternals method), 177
absorb() (framework.node.NodeInternals_Func method), 179
absorb() (framework.node.NodeInternals_GenFunc method), 179
absorb() (framework.node.NodeInternals_NonTerm method), 182
absorb() (framework.node.NodeInternals_Term method), 185
absorb_auto_helper() (framework.node.NodeInternals_Term method), 185
absorb_auto_helper() (framework.node.NodeInternals_TypedValue method), 186
absorb_auto_helper() (framework.value_types.BitField method), 200
absorb_auto_helper() (framework.value_types.INT method), 206
absorb_auto_helper() (framework.value_types.String method), 212
accumulate() (libs.utils.Accumulator method), 302
Accumulator (class in libs.utils), 302
Active (framework.tactics_helpers.DataMakerAttr attribute), 262
Ada (framework.knowledge.information.Language attribute), 295
add() (framework.node.NodeInternals_NonTerm method), 182
add_additional_feedback_interface() (framework.targets.network.NetworkTarget method), 236
add_attributes() (framework.node.NodeSemantics method), 186
add_binding() (framework.targets.debug.TestTarget method), 242
add_conf() (framework.node.Node method), 169
add_extensions() (framework.target_helpers.Target method), 230
add_fbk_from() (framework.knowledge.feedback_collector.FeedbackCollector method), 291
add_feedback_sources() (framework.targets.debug.TestTarget method), 242

add_info() (*framework.data.Data method*), 156
add_information() (*framework.knowledge.information.InformationCollector method*), 295
add_instruction() (*framework.operator_helpers.Operation method*), 246
add_knowledge() (*framework.project.Project method*), 244
add_node_to_corrupt() (*framework.node.Env method*), 164
add_operation() (*framework.data.CallBackOps method*), 156
add_pending_data() (*framework.target_helpers.Target method*), 231
Add_PeriodicData (*framework.data.CallBackOps attribute*), 156
add_probe() (*framework.monitor.Monitor method*), 250
add_specific_fuzzy_vals() (*framework.value_types.INT method*), 206
add_specific_fuzzy_vals() (*framework.value_types.VT method*), 217
add_specific_fuzzy_vals() (*framework.value_types.VT_Alt method*), 218
AddExistingProbeToMonitorError, 249
advance() (*framework.node_builder.RegexParser.Brackets method*), 193
advance() (*framework.node_builder.RegexParser.Brackets.Final method*), 192
advance() (*framework.node_builder.RegexParser.Brackets.Initial method*), 193
advance() (*framework.node_builder.RegexParser.Brackets.Max method*), 193
advance() (*framework.node_builder.RegexParser.Brackets.Min method*), 193
advance() (*framework.node_builder.RegexParser.Escape method*), 194
advance() (*framework.node_builder.RegexParser.Final method*), 194
advance() (*framework.node_builder.RegexParser.Group method*), 194
advance() (*framework.node_builder.RegexParser.Initial method*), 195
advance() (*framework.node_builder.RegexParser.Main method*), 195
advance() (*framework.node_builder.RegexParser.Parenthesis method*), 195
advance() (*framework.node_builder.RegexParser.ParenthesisEscape method*), 195
advance() (*framework.node_builder.RegexParser.ParenthesisFinal method*), 196
advance() (*framework.node_builder.RegexParser.ParenthesisInitial method*), 196
advance() (*framework.node_builder.RegexParser.ParenthesisMain method*), 196
advance() (*framework.node_builder.RegexParser.QtyState method*), 196
advance() (*framework.node_builder.RegexParser.SquareBrackets.AfterRa method*), 197
advance() (*framework.node_builder.RegexParser.SquareBrackets.BeforeR method*), 197
advance() (*framework.node_builder.RegexParser.SquareBrackets.EscapeA method*), 197
advance() (*framework.node_builder.RegexParser.SquareBrackets.EscapeE method*), 197
advance() (*framework.node_builder.RegexParser.SquareBrackets.Final method*), 198
advance() (*framework.node_builder.RegexParser.SquareBrackets.Initial method*), 198
advance() (*framework.node_builder.RegexParser.SquareBrackets.Range method*), 198
advance() (*framework.node_builder.State method*), 199
after_enabling_mode() (*framework.value_types.BitField method*), 200
after_enabling_mode() (*framework.value_types.VT_Alt method*), 218
after_target_feedback_retrieval (*framework.monitor.BlockingProbeUser property*), 250
alt_cformat (*framework.value_types.INT attribute*), 206
alt_cformat (*framework.value_types.SINT16_be attribute*), 209
alt_cformat (*framework.value_types.SINT16_le attribute*), 209
alt_cformat (*framework.value_types.SINT32_be attribute*), 209
alt_cformat (*framework.value_types.SINT32_le attribute*), 210
alt_cformat (*framework.value_types.SINT64_be attribute*), 210
alt_cformat (*framework.value_types.SINT64_le attribute*), 210
alt_cformat (*framework.value_types.SINT8 attribute*), 210
alt_cformat (*framework.value_types.UINT16_be attribute*), 215
alt_cformat (*framework.value_types.UINT16_le attribute*), 215
alt_cformat (*framework.value_types.UINT32_be attribute*), 216
alt_cformat (*framework.value_types.UINT32_le attribute*), 216
alt_cformat (*framework.value_types.UINT64_be attribute*), 216
alt_cformat (*framework.value_types.UINT64_le attribute*), 217
alt_cformat (*framework.value_types.UINT8 attribute*),

217
AltConfConsumer (*class in framework.fuzzing_primitives*), 266
anchor (*framework.scenario.Scenario property*), 278
Android (*framework.knowledge.information.OS attribute*), 295
append_new_process() (*framework.data.DataProcess method*), 159
append_to_alphabet() (*framework.node_builder.RegexParser method*), 199
append_to_buffer() (*framework.node_builder.RegexParser method*), 199
append_to_contents() (*framework.node_builder.RegexParser method*), 199
ARM (*framework.knowledge.information.Hardware attribute*), 294
arm() (*framework.monitor.Probe method*), 251
ASCII (*framework.dmhelpers.generic.MH.Charset attribute*), 285
ASCII (*framework.value_types.String attribute*), 211
ASCII_EXT (*framework.dmhelpers.generic.MH.Charset attribute*), 285
ASK_PASSWORD (*framework.comm_backends.SSH_Backend attribute*), 258
ASK_PASSWORD (*framework.targets.ssh.SSHTarget attribute*), 239
atom_copy() (*framework.data_model.NodeBackend method*), 163
atom_identifiers() (*framework.data_model.DataModel method*), 161
AttrGroup (*class in framework.data*), 155
AutoSeparator (*framework.node.NodeInternals attribute*), 176

B

Backend (*class in framework.comm_backends*), 257
backend (*framework.monitor.ProbeCmd attribute*), 252, 253
backend (*framework.monitor.ProbeMem attribute*), 253, 254
backend (*framework.monitor.ProbePID attribute*), 255
BackendError, 258
BasicVisitor (*class in framework.fuzzing_primitives*), 267
BigEndian (*framework.value_types.VT attribute*), 217
bind_info() (*framework.data.Data method*), 156
bit_length (*framework.value_types.BitField property*), 200
BitField (*class in framework.value_types*), 200
BitFieldCondition (*class in framework.node*), 163
BitReverse_Enc (*class in framework.encoders*), 271
blocking_methods (*framework.monitor.ProbeTimeoutError property*), 256
blocking_probe() (*in module framework.monitor*), 257
BlockingProbeUser (*class in framework.monitor*), 249
branch_to_reinit() (*framework.scenario.Scenario method*), 278
buffer (*framework.node_builder.RegexParser property*), 199
build() (*framework.evolutionary_helpers.EvolutionaryScenariosFactory static method*), 290
build_data_model() (*framework.data_model.DataModel method*), 161
byte_length (*framework.value_types.BitField property*), 200

C

C (*framework.knowledge.information.Language attribute*), 295
c (*framework.node.Node property*), 169
calc_parity_bit() (*in module framework.basic_primitives*), 155
CallBackOps (*class in framework.data*), 155
cancel_absorb() (*framework.node.NodeInternals_Func method*), 179
cancel_absorb() (*framework.node.NodeInternals_GenFunc method*), 179
cancel_absorb() (*framework.node.NodeInternals_NonTerm method*), 183
cancel_absorb() (*framework.node.NodeInternals_Term method*), 185
cc (*framework.node.Node property*), 169
cformat (*framework.value_types.INT attribute*), 206
cformat (*framework.value_types.SINT16_be attribute*), 209
cformat (*framework.value_types.SINT16_le attribute*), 209
cformat (*framework.value_types.SINT32_be attribute*), 209
cformat (*framework.value_types.SINT32_le attribute*), 210
cformat (*framework.value_types.SINT64_be attribute*), 210
cformat (*framework.value_types.SINT64_le attribute*), 210
cformat (*framework.value_types.SINT8 attribute*), 210
cformat (*framework.value_types.UINT16_be attribute*), 215

cformat (*framework.value_types.UINT16_le attribute*), 215
cformat (*framework.value_types.UINT32_be attribute*), 216
cformat (*framework.value_types.UINT32_le attribute*), 216
cformat (*framework.value_types.UINT64_be attribute*), 216
cformat (*framework.value_types.UINT64_le attribute*), 217
cformat (*framework.value_types.UINT8 attribute*), 217
change_subfield() (*framework.value_types.BitField method*), 200
change_subnodes_csts() (*framework.node.NodeInternals_NonTerm method*), 183
check() (*framework.node.BitFieldCondition method*), 163
check() (*framework.node.IntCondition method*), 166
check() (*framework.node.NodeCondition method*), 175
check() (*framework.node.RawCondition method*), 188
check() (*framework.node.SyncExistenceObj method*), 189
check_data_existence() (*framework.database.Database method*), 274
chunk_lines() (*in module libs.utils*), 302
CHUNK_SZ (*framework.targets.network.NetworkTarget attribute*), 233
cleanup() (*framework.data_model.DataModel method*), 161
cleanup() (*framework.knowledge.feedback_collector.FeedbackCollector method*), 291
cleanup() (*framework.scenario.Step method*), 280
cleanup() (*framework.tactics_helpers.Disruptor method*), 262
cleanup() (*framework.tactics_helpers.DynGeneratorFromScenario method*), 263
cleanup() (*framework.tactics_helpers.Generator method*), 264
cleanup() (*framework.tactics_helpers.StatefulDisruptor method*), 264
cleanup() (*framework.target_helpers.Target method*), 231
cleanup() (*framework.targets.local.LocalTarget method*), 238
cleanup() (*framework.targets.network.NetworkTarget method*), 236
cleanup() (*libs.utils.Task method*), 302
cleanup_all_callbacks() (*framework.data.Data method*), 156
cleanup_all_dmakers() (*framework.plumbing.FmkPlumbing method*), 298
cleanup_basic_djobs() (*framework.node.Env method*), 164
cleanup_callbacks() (*framework.data.Data method*), 156
cleanup_dmaker() (*framework.plumbing.FmkPlumbing method*), 298
cleanup_remaining_djobs() (*framework.node.Env method*), 164
CleanupDMakers (*framework.operator_helpers.Operation attribute*), 246
clear() (*framework.data.AttrGroup method*), 155
clear() (*libs.utils.Accumulator method*), 302
clear_all_exhausted_nodes() (*framework.node.Env method*), 164
clear_attr() (*framework.node.Node method*), 169
clear_attr() (*framework.node.NodeInternals method*), 177
clear_attr() (*framework.tactics_helpers.Disruptor method*), 262
clear_attr() (*framework.tactics_helpers.Generator method*), 264
clear_attr() (*framework.tactics_helpers.StatefulDisruptor method*), 264
clear_child_attr() (*framework.node.NodeInternals method*), 177
clear_child_attr() (*framework.node.NodeInternals_GenFunc method*), 180
clear_child_attr() (*framework.node.NodeInternals_NonTerm method*), 183
clear_clone_info_since() (*framework.node.NodeInternals method*), 177
clear_clone_info_since() (*framework.node.NodeInternals_Func method*), 179
clear_clone_info_since() (*framework.node.NodeInternals_GenFunc method*), 180
clear_clone_info_since() (*framework.node.NodeInternals_NonTerm method*), 183
clear_disruptor_clones() (*framework.tactics_helpers.Tactics method*), 265
clear_dmaker_reset() (*framework.scenario.Step method*), 280
clear_drawn_node_attrs() (*framework.node.Env4NT method*), 165
clear_exhausted_node() (*framework.node.Env method*), 164
clear_generator_clones() (*framework.tactics_helpers.Tactics method*), 265
clear_graph_info_since() (*framework.node.DynNode_Helpers method*), 164

clear_items() (*framework.node.NodeCustomization method*), 176
clear_node_constraint() (*framework.node.NodeInternalsCriteria method*), 178
clone() (*framework.scenario.Scenario method*), 278
clone_disruptor() (*framework.tactics_helpers.Tactics method*), 265
clone_ext_node_args_mode (*framework.node.FuncCusto property*), 165
clone_ext_node_args_mode (*framework.node.GenFuncCusto property*), 166
clone_generator() (*framework.tactics_helpers.Tactics method*), 265
CloneExtNodeArgs (*framework.dmhelpers.generic.MH.Custo.Func attribute*), 285
CloneExtNodeArgs (*framework.dmhelpers.generic.MH.Custo.Gen attribute*), 285
CloneExtNodeArgs (*framework.node.FuncCusto attribute*), 165
CloneExtNodeArgs (*framework.node.GenFuncCusto attribute*), 165
collapse_padding_mode (*framework.node.NonTermCusto property*), 188
CollapsePadding (*framework.dmhelpers.generic.MH.Custo.NTerm attribute*), 285
CollapsePadding (*framework.node.NonTermCusto attribute*), 188
collect_data() (*framework.knowledge.feedback_handler.FeedbackHandler method*), 292
collect_feedback() (*framework.logger.Logger method*), 248
collect_residual_feedback() (*framework.plumbing.FmkPlumbing method*), 298
collect_unsolicited_feedback() (*framework.target_helpers.Target method*), 231
collect_unsolicited_feedback() (*framework.targets.network.NetworkTarget method*), 236
color_enabled (*framework.node.Env property*), 164
color_enabled (*framework.node.Node property*), 169
column_names_from() (*framework.database.Database method*), 274
command_pattern (*framework.monitor.ProbeMem attribute*), 254
command_pattern (*framework.monitor.ProbePID attribute*), 255
comment (*framework.dmhelpers.xml.TAG_TYPE attribute*), 288
commit_data_table_entry() (*framework.logger.Logger method*), 248
compliant_with() (*framework.node.Node method*), 169
compute_sub_graphs() (*framework.evolutionary_helpers.CrossoverHelper.Operand method*), 289
conf() (*framework.node.Node method*), 169
configure() (*framework.monitor.Probe method*), 251
configure_probe() (*framework.monitor.Monitor method*), 250
confirm_absorb() (*framework.node.NodeInternals_Func method*), 179
confirm_absorb() (*framework.node.NodeInternals_GenFunc method*), 180
confirm_absorb() (*framework.node.NodeInternals_NonTerm method*), 183
confirm_absorb() (*framework.node.NodeInternals_Term method*), 185
confs (*framework.node.Node property*), 169
connect_to() (*framework.scenario.Step method*), 280
connect_to() (*framework.targets.network.NetworkTarget method*), 236
Constraint (*class in framework.constraint_helpers*), 297
ConstraintError, 298
consume_node() (*framework.fuzzing_primitives.AltConfConsumer method*), 266
consume_node() (*framework.fuzzing_primitives.BasicVisitor method*), 267
consume_node() (*framework.fuzzing_primitives.NodeConsumerStub method*), 268
consume_node() (*framework.fuzzing_primitives.NonTermVisitor method*), 269
consume_node() (*framework.fuzzing_primitives.SeparatorDisruption method*), 270
consume_node() (*framework.fuzzing_primitives.TypedNodeDisruption method*), 270
consumer_start (*framework.targets.debug.TestTarget attribute*), 242
consumer_stop (*framework.targets.debug.TestTarget attribute*), 242
content (*framework.data.Data property*), 156
content (*framework.data.DataBackend property*), 158

content (*framework.data.EmptyBackend property*), 159
content (*framework.data.NodeBackend property*), 159
content (*framework.data.RawBackend property*), 160
content (*framework.scenario.Step property*), 280
Controller (*framework.tactics_helpers.DataMakerAttr attribute*), 262
Copy (*framework.dmhelpers.generic.MH attribute*), 285
copy_attrs_from() (*framework.value_types.INT method*), 206
copy_attrs_from() (*framework.value_types.INT_str method*), 208
copy_attrs_from() (*framework.value_types.VT method*), 217
copy_callback_from() (*framework.data.Data method*), 156
copy_from() (*framework.data.AttrGroup method*), 155
copy_from() (*framework.node.NodeCustomization method*), 176
COPY_VALUE() (*in module framework.dmhelpers.generic*), 283
corrupt_bits() (*in module framework.basic_primitives*), 155
corrupt_bytes() (*in module framework.basic_primitives*), 155
CORRUPT_EXIST_COND (*framework.node.Node attribute*), 167
CORRUPT_NODE_QTY (*framework.node.Node attribute*), 167
CORRUPT_QTY_SYNC (*framework.node.Node attribute*), 167
CORRUPT_SIZE_SYNC (*framework.node.Node attribute*), 167
count_of_possible_values (*framework.value_types.BitField property*), 201
CRC() (*in module framework.dmhelpers.generic*), 283
create_atom_from_raw_data() (*framework.data_model.DataModel method*), 161
create_graph_from_desc() (*framework.node_builder.NodeBuilder method*), 192
crossover_algo1() (*framework.evolutionary_helpers.CrossoverHelper class method*), 289
CrossoverHelper (*class in framework.evolutionary_helpers*), 289
CrossoverHelper.Operand (*class in framework.evolutionary_helpers*), 289
CSP (*class in framework.constraint_helpers*), 296
Ctrl_Char_Set (*framework.knowledge.information.InputHandling attribute*), 295
ctrl_char_set (*framework.value_types.String attribute*), 212
current_conf (*framework.node.Node attribute*), 166
current_qty (*framework.node.NodeInternals_NonTerm.NodeAttrs property*), 181
current_step (*framework.scenario.Scenario property*), 278
Cursory (*framework.knowledge.information.Test attribute*), 296
customize() (*framework.node.NodeInternals method*), 177
customize() (*framework.node.NodeInternals_Func method*), 179
customize_node_backend() (*framework.data_model.DataModel method*), 161
CYCLE() (*in module framework.dmhelpers.generic*), 284
cycle_clone_mode (*framework.node.NonTermCusto property*), 188
CycleClone (*framework.dmhelpers.generic.MH.Custo.NTerm attribute*), 285
CycleClone (*framework.node.NonTermCusto attribute*), 188

D

d_add_data (*class in framework.generic_data_makers*), 219
d_call_external_program (*class in framework.generic_data_makers*), 219
d_call_function (*class in framework.generic_data_makers*), 219
d_corrupt_bits_by_position (*class in framework.generic_data_makers*), 220
d_corrupt_node_bits (*class in framework.generic_data_makers*), 220
d_fix_constraints (*class in framework.generic_data_makers*), 220
d_fuzz_model_structure (*class in framework.generic_data_makers*), 221
d_max_size (*class in framework.generic_data_makers*), 221
d_modify_nodes (*class in framework.generic_data_makers*), 221
d_next_node_content (*class in framework.generic_data_makers*), 222
d_operate_on_nodes (*class in framework.generic_data_makers*), 222
d_shallow_copy (*class in framework.generic_data_makers*), 223
d_switch_to_alternate_conf (*class in framework.generic_data_makers*), 223
Data (*class in framework.data*), 156
data_desc (*framework.scenario.Step property*), 280
data_id (*framework.tactics_helpers.dyn_generator attribute*), 266
data_id (*framework.tactics_helpers.DynGenerator attribute*), 262

data_maker_name (*framework.data.DataBackend property*), 158
data_maker_name (*framework.data.EmptyBackend property*), 159
data_maker_name (*framework.data.NodeBackend property*), 159
data_maker_name (*framework.data.RawBackend property*), 160
data_maker_type (*framework.data.DataBackend property*), 158
data_maker_type (*framework.data.EmptyBackend property*), 159
data_maker_type (*framework.data.NodeBackend property*), 160
data_maker_type (*framework.data.RawBackend property*), 160
data_model (*framework.data.DataBackend property*), 158
data_model (*framework.data.EmptyBackend property*), 159
data_start (*framework.targets.debug.TestTarget attribute*), 242
DataAttr (*class in framework.data*), 157
DataBackend (*class in framework.data*), 158
Database (*class in framework.database*), 273
DataMaker (*class in framework.tactics_helpers*), 261
DataMakerAttr (*class in framework.tactics_helpers*), 262
DataModel (*class in framework.data_model*), 160
DataProcess (*class in framework.data*), 158
DDL_fname (*framework.database.Database attribute*), 273
DEBUG (*framework.dmhelpers.generic.MH.Attr attribute*), 284
debug (*framework.node.Node property*), 169
DEBUG (*framework.node.NodeInternals attribute*), 176
debug (*framework.node.NodeInternals property*), 177
decode() (*framework.data_model.DataModel method*), 161
decode() (*framework.encoders.BitReverse_Enc method*), 271
decode() (*framework.encoders.Encoder method*), 271
decode() (*framework.encoders.GSM7bitPacking_Enc method*), 272
decode() (*framework.encoders.GSMPhoneNum_Enc method*), 272
decode() (*framework.encoders.GZIP_Enc method*), 272
decode() (*framework.encoders.Wrap_Enc method*), 273
decode() (*framework.value_types.String method*), 213
decrease_trust() (*framework.knowledge.information.Info method*), 294
Deep (*framework.knowledge.information.Test attribute*), 296
default_custo (*framework.node.NodeInternals attribute*), 177
default_custo (*framework.node.NodeInternals_Func attribute*), 179
default_custo (*framework.node.NodeInternals_GenFunc attribute*), 180
default_custo (*framework.node.NodeInternals_NonTerm attribute*), 183
DEFAULT_DB_NAME (*framework.database.Database attribute*), 273
DEFAULT_DISABLED_NODEINT (*framework.node.Node attribute*), 167
DEFAULT_DISABLED_VALUE (*framework.node.Node attribute*), 167
default_dm (*framework.project.Project attribute*), 244
DEFAULT_DM_NAME (*framework.database.Database attribute*), 273
default_gen_custo (*framework.data_model.NodeBackend attribute*), 163
DEFAULT_GEN_NAME (*framework.database.Database attribute*), 273
DEFAULT_GTYPE_NAME (*framework.database.Database attribute*), 273
DEFAULT_MAX_SZ (*framework.value_types.String attribute*), 211
default_nonterm_custo (*framework.data_model.NodeBackend attribute*), 163
default_qty (*framework.node.NodeInternals_NonTerm.NodeAttrs property*), 181
DefaultIndividual (*class in framework.evolutionary_helpers*), 289
DefaultPopulation (*class in framework.evolutionary_helpers*), 289
del_extensions() (*framework.target_helpers.Target method*), 231
Del_PeriodicData (*framework.data.CallBackOps attribute*), 156
delay (*framework.monitor.Probe property*), 252
delay (*framework.monitor.ProbePID attribute*), 255
delay_between_attempts (*framework.monitor.ProbePID attribute*), 255
delay_collapsing (*framework.node.NonTermCusto property*), 188
DelayCollapsing (*framework.dmhelpers.generic.MH.Custo.NTerm attribute*), 285
DelayCollapsing (*framework.node.NonTermCusto attribute*), 188
delayed_jobs_pending (*framework.node.Env property*), 164

depth (*framework.node.Node attribute*), 167
Determinist (*framework.dmhelpers.generic.MH.Attr attribute*), 284
Determinist (*framework.knowledge.information.OperationMode attribute*), 295
determinist (*framework.node.DynNode_Helpers attribute*), 164
Determinist (*framework.node.NodeInternals attribute*), 176
determinist (*framework.value_types.INT attribute*), 206
disable() (*framework.database.Database method*), 274
disable_color() (*framework.node.Env method*), 164
disable_color() (*framework.node.Node method*), 169
disable_feedback_handlers() (*framework.project.Project method*), 244
disable_fmkdb() (*framework.plumbing.FmkPlumbing method*), 298
disable_hooks() (*framework.monitor.Monitor method*), 250
disable_wkspace() (*framework.plumbing.FmkPlumbing method*), 298
DISABLED (*framework.node.NodeInternals attribute*), 176
disp (*libs.utils.ExternalDisplay property*), 302
display_color_theme() (*framework.plumbing.FmkPlumbing method*), 298
display_data_info() (*framework.database.Database method*), 274
display_data_info_by_date() (*framework.database.Database method*), 274
display_data_info_by_range() (*framework.database.Database method*), 274
display_feedback (*framework.knowledge.feedback_collector.FeedbackSource property*), 292
display_feedback (*framework.target_helpers.Target attribute*), 231
display_stats() (*framework.database.Database method*), 274
disrupt_data() (*framework.generic_data_makers.d_add_data method*), 219
disrupt_data() (*framework.generic_data_makers.d_call_external_program method*), 219
disrupt_data() (*framework.generic_data_makers.d_call_function method*), 220
disrupt_data() (*framework.generic_data_makers.d_corrupt_bits_by_position method*), 220
disrupt_data() (*framework.generic_data_makers.d_corrupt_node_bits method*), 220
disrupt_data() (*framework.generic_data_makers.d_fix_constraints method*), 221
disrupt_data() (*framework.generic_data_makers.d_fuzz_model_structure method*), 221
disrupt_data() (*framework.generic_data_makers.d_max_size method*), 221
disrupt_data() (*framework.generic_data_makers.d_modify_nodes method*), 222
disrupt_data() (*framework.generic_data_makers.d_next_node_content method*), 222
disrupt_data() (*framework.generic_data_makers.d_operate_on_nodes method*), 223
disrupt_data() (*framework.generic_data_makers.d_shallow_copy method*), 223
disrupt_data() (*framework.generic_data_makers.d_switch_to_alternate_conf method*), 223
disrupt_data() (*framework.generic_data_makers.sd_constraint_fuzz method*), 225
disrupt_data() (*framework.generic_data_makers.sd_fuzz_separator_nodes method*), 225
disrupt_data() (*framework.generic_data_makers.sd_fuzz_typed_nodes method*), 226
disrupt_data() (*framework.generic_data_makers.sd_struct_constraints method*), 227
disrupt_data() (*framework.generic_data_makers.sd_switch_to_alternate_conf method*), 227
disrupt_data() (*framework.generic_data_makers.sd_walk_csp_solutions method*), 228
disrupt_data() (*framework.generic_data_makers.sd_walk_data_model method*), 229
disrupt_data() (*framework.tactics_helpers.Disruptor method*), 262
disrupt_data() (*framework.tactics_helpers.StatefulDisruptor method*), 264
Disruptor (*class in framework.tactics_helpers*), 262
disruptor() (*in module framework.tactics_helpers*), 266

disruptor_types (*framework.tactics_helpers.Tactics property*), 265
disruptors_info() (*framework.tactics_helpers.Tactics method*), 265
DJobGroup (*class in framework.node*), 163
djobs_exists() (*framework.node.Env method*), 164
DJOBS_PRIO_dynhelpers (*framework.node.Node attribute*), 167
DJOBS_PRIO_genfunc (*framework.node.Node attribute*), 167
DJOBS_PRIO_nterm_existence (*framework.node.Node attribute*), 167
dlen_format (*framework.targets.debug.TestTarget attribute*), 242
dlen_start (*framework.targets.debug.TestTarget attribute*), 242
dlen_stop (*framework.targets.debug.TestTarget attribute*), 242
dm (*framework.scenario.ScenarioEnv property*), 279
dm (*libs.utils.Task attribute*), 302
do_absorb() (*framework.node.NodeInternals_Term method*), 185
do_absorb() (*framework.node.NodeInternals_TypedValue method*), 186
do_absorb() (*framework.value_types.BitField method*), 201
do_absorb() (*framework.value_types.INT method*), 206
do_absorb() (*framework.value_types.String method*), 213
do_after_all() (*framework.operator_helpers.Operator method*), 246
do_after_reset() (*framework.fuzzing_primitives.NodeConsumerStub method*), 268
do_before_data_processing() (*framework.scenario.Step method*), 280
do_before_sending() (*framework.scenario.Step method*), 280
do_cleanup_absorb() (*framework.node.NodeInternals_Term method*), 185
do_cleanup_absorb() (*framework.node.NodeInternals_TypedValue method*), 186
do_cleanup_absorb() (*framework.value_types.BitField method*), 201
do_cleanup_absorb() (*framework.value_types.INT method*), 206
do_cleanup_absorb() (*framework.value_types.String method*), 213
do_revert_absorb() (*framework.node.NodeInternals_Term method*), 185
do_revert_absorb() (*framework.node.NodeInternals_TypedValue method*), 186
do_revert_absorb() (*framework.value_types.BitField method*), 201
do_revert_absorb() (*framework.value_types.INT method*), 206
do_revert_absorb() (*framework.value_types.String method*), 213
dyn_generator (*class in framework.tactics_helpers*), 266
dyn_generator_from_scenario (*class in framework.tactics_helpers*), 266
dynamic_generator_ids() (*framework.plumbing.FmkPlumbing method*), 298
DynGenerator (*class in framework.tactics_helpers*), 262
DynGeneratorFromScenario (*class in framework.tactics_helpers*), 263
DynNode_Helpers (*class in framework.node*), 163

E

empty_data_bank() (*framework.plumbing.FmkPlumbing method*), 298
empty_workspace() (*framework.plumbing.FmkPlumbing method*), 298
EmptyBackend (*class in framework.data*), 159
EmptyDataProcess (*class in framework.data*), 159
EmptyTarget (*class in framework.target_helpers*), 229
enable() (*framework.database.Database method*), 274
enable_color() (*framework.node.Env method*), 164
enable_color() (*framework.node.Node method*), 169
enable_feedback_handlers() (*framework.project.Project method*), 244
enable_fmkdb() (*framework.plumbing.FmkPlumbing method*), 298
enable_fuzz_mode() (*framework.value_types.VT_Alt method*), 218
enable_hooks() (*framework.monitor.Monitor method*), 250
enable_normal_mode() (*framework.value_types.VT_Alt method*), 218
enable_wkspace() (*framework.plumbing.FmkPlumbing method*), 298
enc2struct (*framework.value_types.VT attribute*), 217
encode() (*framework.encoders.BitReverse_Enc method*), 271
encode() (*framework.encoders.Encoder method*), 271
encode() (*framework.encoders.GSM7bitPacking_Enc method*), 272
encode() (*framework.encoders.GSMPhoneNum_Enc method*), 272
encode() (*framework.encoders.GZIP_Enc method*), 272
encode() (*framework.encoders.Wrap_Enc method*), 273
encode() (*framework.value_types.String method*), 213

encoded_string (*framework.value_types.String attribute*), 211, 213
Encoder (*class in framework.encoders*), 271
encoding_test_cases() (*framework.value_types.String method*), 213
endian (*framework.value_types.INT attribute*), 206
endian (*framework.value_types.INT_str attribute*), 208
endian (*framework.value_types.SINT16_be attribute*), 209
endian (*framework.value_types.SINT16_le attribute*), 209
endian (*framework.value_types.SINT32_be attribute*), 209
endian (*framework.value_types.SINT32_le attribute*), 210
endian (*framework.value_types.SINT64_be attribute*), 210
endian (*framework.value_types.SINT64_le attribute*), 210
endian (*framework.value_types.SINT8 attribute*), 210
endian (*framework.value_types.UINT16_be attribute*), 215
endian (*framework.value_types.UINT16_le attribute*), 215
endian (*framework.value_types.UINT32_be attribute*), 216
endian (*framework.value_types.UINT32_le attribute*), 216
endian (*framework.value_types.UINT64_be attribute*), 216
endian (*framework.value_types.UINT64_le attribute*), 217
endian (*framework.value_types.UINT8 attribute*), 217
endian (*framework.value_types.VT attribute*), 217
enforce_absorb_constraints() (*framework.node.Node method*), 170
enforce_absorb_constraints() (*framework.node.NodeInternals method*), 177
entangle_with() (*framework.node.Node method*), 170
entangled_nodes (*framework.node.Node attribute*), 167
Env (*class in framework.node*), 164
env (*framework.node.Node attribute*), 166
env (*framework.node.NodeInternals property*), 177
env (*framework.node.NodeInternals_GenFunc property*), 180
env (*framework.scenario.Scenario property*), 278
Env4NT (*class in framework.node*), 165
estimate_last_data_impact_uniqueness() (*framework.knowledge.feedback_handler.FeedbackHandler method*), 292
estimate_last_data_impact_uniqueness() (*framework.project.Project method*), 244
EvolutionaryScenariosFactory (*class in framework.evolutionary_helpers*), 290
evolve() (*framework.evolutionary_helpers.DefaultPopulation method*), 290
evolve() (*framework.evolutionary_helpers.Population method*), 291
exec_command() (*framework.comm_backends.Backend method*), 257
exec_dm_tests() (*framework.plumbing.FmkPlumbing method*), 298
execute_basic_djobs() (*framework.node.Env method*), 164
execute_sql_statement() (*framework.database.Database method*), 274
exhausted_node_exists() (*framework.node.Env method*), 164
exhausted_nodes_amount() (*framework.node.Env method*), 164
exhausted_seq (*framework.node.NodeInternals_NonTerm.NodeAttrs attribute*), 181
exhausted_solutions (*framework.constraint_helpers.CSP property*), 297
Existence (*framework.node.SyncScope attribute*), 189
existence_corrupt_hook() (*framework.node.NodeInternals_NonTerm static method*), 183
export_data() (*framework.database.Database method*), 274
Exportable (*framework.operator_helpers.Operation attribute*), 246
ExportableFMKOps (*class in framework.plumbing*), 298
extend() (*framework.node.NodeInternalsCriteria method*), 178
extend() (*framework.node.NodeSemanticsCriteria method*), 187
extend() (*framework.value_types.BitField method*), 201
extend_left() (*framework.value_types.BitField method*), 201
extend_right() (*framework.value_types.BitField method*), 201
extended_char_set (*framework.value_types.String attribute*), 213
extensions (*framework.target_helpers.Target property*), 231
ExternalDisplay (*class in libs.utils*), 302
extract_info_from_feedback() (*framework.knowledge.feedback_handler.FeedbackHandler method*), 292
extract_info_from_feedback() (*framework.knowledge.feedback_handler.TestFbkHandler method*), 294

F

fbk_lock (*framework.knowledge.feedback_collector.FeedbackCollector attribute*), 291
FBK_WAIT_FULL_TIME (*framework.target_helpers.Target attribute*), 230
fbk_wait_full_time_slot_mode (*framework.target_helpers.Target property*), 231
fbk_wait_full_time_slot_msg (*framework.target_helpers.Target attribute*), 231
FBK_WAIT_UNTIL_RECV (*framework.target_helpers.Target attribute*), 230
fbk_wait_until_recv_mode (*framework.target_helpers.Target property*), 231
fbk_wait_until_recv_msg (*framework.target_helpers.Target attribute*), 231
feedback_gate (*libs.utils.Task attribute*), 302
feedback_mode (*framework.scenario.Step property*), 280
feedback_timeout (*framework.scenario.Step property*), 280
feedback_timeout (*framework.target_helpers.Target attribute*), 231
FEEDBACK_TRAIL_TIME_WINDOW (*framework.database.Database attribute*), 273
FeedbackCollector (*class in framework.knowledge.feedback_collector*), 291
FeedbackGate (*class in framework.database*), 275
FeedbackHandler (*class in framework.knowledge.feedback_handler*), 292
FeedbackSource (*class in framework.knowledge.feedback_collector*), 292
fetch_data() (*framework.database.Database method*), 274
file_extension (*framework.data_model.DataModel attribute*), 162
Filename (*class in framework.value_types*), 202
filter_out_entangled_nodes() (*framework.node.Node static method*), 170
FinalStep (*class in framework.scenario*), 276
find_file() (*in module libs.utils*), 303
Finite (*framework.dmhelpers.generic.MH.Attr attribute*), 284
Finite (*framework.node.NodeInternals attribute*), 176
fix_synchronized_nodes() (*framework.node.Node method*), 170
flatten() (*in module framework.node*), 190
flatten_node_list() (*framework.node.NodeInternals_NonTerm method*), 183
flush() (*framework.logger.Logger method*), 248
flush() (*framework.node_builder.RegexParser method*), 199
flush() (*framework.plumbing.Printer method*), 301
FLUSH_API (*framework.logger.Logger attribute*), 247
flush_collector() (*framework.knowledge.feedback_handler.FeedbackHandler method*), 293
flush_current_feedback() (*framework.database.Database method*), 274
flush_errors() (*framework.plumbing.FmkPlumbing method*), 298
flush_feedback() (*framework.database.Database method*), 274
fmkDB (*framework.logger.Logger attribute*), 248
fmkdb_fetch_data() (*framework.plumbing.FmkPlumbing method*), 298
fmkops (*libs.utils.Task attribute*), 302
FmkPlumbing (*class in framework.plumbing*), 298
FmkTask (*class in framework.plumbing*), 301
FolderPath (*class in framework.value_types*), 204
ForceDataHandling (*framework.data.CallBackOps attribute*), 156
formatted_str() (*framework.data.DataProcess method*), 159
forward_conf_change_mode (*framework.node.GenFuncCusto property*), 166
ForwardConfChange (*framework.dmhelpers.generic.MH.Custo.Gen attribute*), 285
ForwardConfChange (*framework.node.GenFuncCusto attribute*), 165
framework.basic_primitives
    module, 155
framework.comm_backends
    module, 257
framework.constraint_helpers
    module, 296
framework.data
    module, 155
framework.data_model
    module, 160
framework.database
    module, 273
framework.dmhelpers.generic
    module, 283
framework.dmhelpers.xml
    module, 288
framework.encoders
    module, 271
framework.evolutionary_helpers
    module, 289
framework.fuzzing_primitives
    module, 266
framework.generic_data_makers
    module, 219
framework.knowledge.feedback_collector
    module, 291
framework.knowledge.feedback_handler

module, 292
framework.knowledge.information
module, 294
framework.logger
module, 247
framework.monitor
module, 249
framework.node
module, 163
framework.node_builder
module, 190
framework.operator_helpers
module, 245
framework.plumbing
module, 298
framework.project
module, 244
framework.scenario
module, 276
framework.tactics_helpers
module, 261
framework.target_helpers
module, 229
framework.targets.debug
module, 242
framework.targets.local
module, 238
framework.targets.network
module, 233
framework.targets.printer
module, 241
framework.targets.ssh
module, 239
framework.value_types
module, 200
Freezable (*framework.dmhelpers.generic.MH.Attr attribute*), 284
Freezable (*framework.node.NodeInternals attribute*), 176
freeze() (*framework.node.Node method*), 170
from_encoder() (*in module framework.value_types*), 219
from_var_to_varns() (*framework.constraint_helpers.CSP method*), 297
frozen_args_mode (*framework.node.FuncCusto property*), 165
frozen_copy_mode (*framework.node.NonTermCusto property*), 188
FrozenArgs (*framework.dmhelpers.generic.MH.Custo.Func attribute*), 285
FrozenArgs (*framework.node.FuncCusto attribute*), 165
FrozenCopy (*framework.dmhelpers.generic.MH.Custo.NTerm attribute*), 285
FrozenCopy (*framework.node.NonTermCusto attribute*), 188
full_combinatory_mode (*framework.node.NonTermCusto property*), 188
FullCombinatory (*framework.dmhelpers.generic.MH.Custo.NTerm attribute*), 285
FullCombinatory (*framework.node.NonTermCusto attribute*), 188
FullyRandom (*framework.dmhelpers.generic.MH attribute*), 285
FuncCusto (*class in framework.node*), 165
fuzz_cases_c_strings() (*framework.value_types.String static method*), 213
fuzz_cases_ctrl_chars() (*framework.value_types.String static method*), 214
fuzz_cases_letter_case() (*framework.value_types.String static method*), 214
fuzz_data_tree() (*in module framework.fuzzing_primitives*), 270
fuzz_weight (*framework.node.Node attribute*), 167
fuzzy_values (*framework.value_types.INT attribute*), 206
fuzzy_values (*framework.value_types.INT16 attribute*), 207
fuzzy_values (*framework.value_types.INT32 attribute*), 207
fuzzy_values (*framework.value_types.INT64 attribute*), 207
fuzzy_values (*framework.value_types.INT8 attribute*), 208
fuzzy_values (*framework.value_types.INT_str attribute*), 208

G

g_generic_pattern (*class in framework.generic_data_makers*), 224
g_population (*class in framework.generic_data_makers*), 224
gather_alt_confs() (*framework.node.Node method*), 170
GEN_MAX_INT (*framework.value_types.INT attribute*), 205
GEN_MIN_INT (*framework.value_types.INT attribute*), 205
General_Info_ID (*framework.targets.network.NetworkTarget attribute*), 233
generate_data() (*framework.generic_data_makers.g_generic_pattern method*), 224

generate_data() (*framework.generic_data_makers.g_population method*), 224
generate_data() (*framework.tactics_helpers.DynGenerator method*), 262
generate_data() (*framework.tactics_helpers.DynGeneratorFromScenario method*), 263
generate_data() (*framework.tactics_helpers.Generator method*), 264
generate_info_from_content() (*framework.data.Data method*), 156
generated_node (*framework.node.NodeInternals_GenFunc property*), 180
Generator (*class in framework.tactics_helpers*), 264
Generator (*framework.dmhelpers.generic.MH attribute*), 285
generator() (*in module framework.tactics_helpers*), 266
generator_types (*framework.tactics_helpers.Tactics property*), 265
generators_info() (*framework.tactics_helpers.Tactics method*), 265
GenFuncCusto (*class in framework.node*), 165
get_all_confs() (*framework.data_model.NodeBackend method*), 163
get_all_constraints() (*framework.constraint_helpers.CSP method*), 297
get_all_djob_groups() (*framework.node.Env method*), 164
get_all_node_constraints() (*framework.node.NodeInternalsCriteria method*), 178
get_all_paths() (*framework.node.Node method*), 170
get_all_paths_from() (*framework.node.Node method*), 170
get_atom() (*framework.data_model.DataModel method*), 162
get_atom_for_absorption() (*framework.data_model.DataModel method*), 162
get_attrs_copy() (*framework.node.NodeInternals method*), 177
get_available_targets() (*framework.plumbing.FmkPlumbing method*), 298
get_basic_djobs() (*framework.node.Env method*), 164
get_bytes() (*framework.knowledge.feedback_collector.FeedbackCollector method*), 291
get_caller_object() (*in module libs.utils*), 303
get_child_all_path() (*framework.node.NodeInternals_GenFunc method*), 180
get_child_all_path() (*framework.node.NodeInternals_NonTerm method*), 183
get_child_all_path() (*framework.node.NodeInternals_Term method*), 185
get_child_nodes_by_attr() (*framework.node.NodeInternals_Empty method*), 178
get_child_nodes_by_attr() (*framework.node.NodeInternals_GenFunc method*), 180
get_child_nodes_by_attr() (*framework.node.NodeInternals_NonTerm method*), 183
get_child_nodes_by_attr() (*framework.node.NodeInternals_Term method*), 185
get_clone() (*framework.node.Node method*), 170
get_comments() (*framework.operator_helpers.LastInstruction method*), 245
get_concrete_nodes() (*framework.node.NodeAbstraction method*), 175
get_configured_crossover_algo2() (*framework.evolutionary_helpers.CrossoverHelper class method*), 289
get_constraint() (*framework.constraint_helpers.CSP method*), 297
get_consumer_idx() (*framework.targets.debug.TestTarget method*), 242
get_content() (*framework.data.Data method*), 156
get_content() (*framework.data.DataBackend method*), 158
get_content() (*framework.data.EmptyBackend method*), 159
get_content() (*framework.data.NodeBackend method*), 160
get_content() (*framework.data.RawBackend method*), 160
get_csp() (*framework.node.Node method*), 171
get_current_conf() (*framework.node.Node method*), 171
get_current_raw_val() (*framework.value_types.BitField method*), 201
get_current_raw_val() (*framework.value_types.INT method*), 206
get_current_raw_val() (*framework.value_types.String method*), 214
get_current_raw_val() (*framework.value_types.VT method*), 217
get_current_subkind() (*framework.node.NodeInternals_GenFunc method*), 180

work.node.NodeInternals method), 177
get_current_subkind() (*framework.node.NodeInternals_TypedValue method*), 186
get_current_value() (*framework.value_types.BitField method*), 201
get_current_value() (*framework.value_types.INT method*), 206
get_current_value() (*framework.value_types.String method*), 214
get_current_value() (*framework.value_types.VT method*), 217
get_data() (*framework.scenario.Step method*), 280
get_data_id() (*framework.data.Data method*), 156
get_data_model() (*framework.data.Data method*), 156
get_data_model() (*framework.node.Env method*), 164
get_data_model_by_name() (*framework.plumbing.FmkPlumbing method*), 298
get_data_models() (*framework.plumbing.FmkPlumbing method*), 298
get_data_with_impact() (*framework.database.Database method*), 274
get_data_with_specific_fbk() (*framework.database.Database method*), 274
get_data_without_fbk() (*framework.database.Database method*), 274
get_datatype_total_weight() (*framework.tactics_helpers.Tactics method*), 265
get_default_db_path() (*framework.database.Database static method*), 274
get_desc() (*framework.scenario.Step method*), 280
get_description() (*framework.target_helpers.Target method*), 231
get_description() (*framework.targets.local.LocalTarget method*), 238
get_description() (*framework.targets.network.NetworkTarget method*), 236
get_description() (*framework.targets.printer.PrinterTarget method*), 241
get_description() (*framework.targets.ssh.SSHTarget method*), 240
get_disruptor_name() (*framework.tactics_helpers.Tactics method*), 265
get_disruptor_obj() (*framework.tactics_helpers.Tactics method*), 265
get_disruptor_validness() (*framework.tactics_helpers.Tactics method*), 265
get_disruptor_weight() (*framework.tactics_helpers.Tactics method*), 265
get_disruptors_list() (*framework.tactics_helpers.Tactics method*), 265
get_djobs_by_gid() (*framework.node.Env method*), 164
get_dmaker_type_total_weight() (*framework.tactics_helpers.Tactics method*), 265
get_drawn_node_qty() (*framework.node.Env4NT method*), 165
get_drawn_node_qty() (*framework.node.NodeInternals_NonTerm method*), 183
get_drawn_node_sz() (*framework.node.Env4NT method*), 165
get_env() (*framework.node.Node method*), 171
get_error() (*framework.plumbing.FmkPlumbing method*), 298
get_error_code() (*framework.knowledge.feedback_collector.FeedbackCollector method*), 291
get_exclusive_criteria() (*framework.node.NodeSemanticsCriteria method*), 187
get_exhausted_nodes() (*framework.node.Env method*), 164
get_external_atom() (*framework.data_model.DataModel method*), 162
get_fbk_mode_desc() (*framework.target_helpers.Target static method*), 231
get_feedback() (*framework.target_helpers.Target method*), 231
get_feedback() (*framework.targets.debug.TestTarget method*), 242
get_feedback() (*framework.targets.local.LocalTarget method*), 238
get_feedback() (*framework.targets.network.NetworkTarget method*), 236
get_feedback_from() (*framework.database.FeedbackGate method*), 275
get_first_node_by_path() (*framework.node.Node method*), 171
get_from_data_bank() (*framework.plumbing.FmkPlumbing method*), 298
get_full_description() (*framework.scenario.Step method*), 280
get_fuzz_weight() (*framework.node.Node method*), 171
get_fuzzed_vt_list() (*framework.value_types.INT method*), 206
get_fuzzed_vt_list() (*framework.value_types.INT_str method*), 208
get_fuzzed_vt_list() (*framework.value_types.VT method*), 218
get_generator_name() (*frame-*

work.tactics_helpers.Tactics method), 265
get_generator_obj() (framework.tactics_helpers.Tactics method), 265
get_generator_validness() (framework.tactics_helpers.Tactics method), 265
get_generator_weight() (framework.tactics_helpers.Tactics method), 265
get_generators_list() (framework.tactics_helpers.Tactics method), 265
get_history() (framework.data.Data method), 156
get_import_directory_path() (framework.data_model.DataModel method), 162
get_info_from_obj() (framework.tactics_helpers.Tactics method), 265
get_initial_dmaker() (framework.data.Data method), 157
get_instructions() (framework.operator_helpers.Operation method), 246
get_internals_backup() (framework.node.Node method), 171
get_last_data() (framework.plumbing.FmkPlumbing method), 299
get_last_target_ack_date() (framework.target_helpers.Target method), 231
get_last_target_ack_date() (framework.targets.debug.TestTarget method), 242
get_last_target_ack_date() (framework.targets.network.NetworkTarget method), 236
get_last_target_ack_date() (framework.targets.ssh.SSHTarget method), 240
get_length() (framework.data.Data method), 157
get_length() (framework.data.DataBackend method), 158
get_length() (framework.data.EmptyBackend method), 159
get_length() (framework.data.NodeBackend method), 160
get_length() (framework.data.RawBackend method), 160
get_mandatory_criteria() (framework.node.NodeSemanticsCriteria method), 187
get_negative_criteria() (framework.node.NodeSemanticsCriteria method), 187
get_next_data_id() (framework.database.Database method), 274
get_node_args() (framework.node.NodeInternals_Func method), 179
get_node_args() (framework.node.NodeInternals_GenFunc method), 180
get_node_constraint() (framework.node.NodeInternalsCriteria method), 178
get_node_containers() (framework.node.SyncExistenceObj method), 189
get_node_containers() (framework.node.SyncObj method), 189
get_node_containers() (framework.node.SyncQtyFromObj method), 189
get_node_containers() (framework.node.SyncSizeObj method), 190
get_node_sync() (framework.node.NodeInternals method), 177
get_nodes_by_paths() (framework.node.Node method), 171
get_nodes_names() (framework.node.Node method), 171
get_operations() (framework.data.CallBackOps method), 156
get_operator() (framework.plumbing.FmkPlumbing method), 299
get_operator() (framework.project.Project method), 244
get_operator_feedback() (framework.operator_helpers.LastInstruction method), 245
get_operator_status() (framework.operator_helpers.LastInstruction method), 245
get_operators() (framework.project.Project method), 244
get_optionalbut1_criteria() (framework.node.NodeSemanticsCriteria method), 187
get_path_from() (framework.node.Node method), 171
get_periodic_description() (framework.scenario.Step method), 280
get_periodic_ref() (framework.scenario.Step method), 280
get_post_args() (framework.targets.local.LocalTarget method), 238
get_pre_args() (framework.targets.local.LocalTarget method), 238
get_printer_name() (framework.targets.printer.PrinterTarget method), 241
get_private() (framework.node.Node method), 171
get_private() (framework.node.NodeInternals method), 177
get_private_info() (framework.monitor.ProbeStatus method), 256
get_probe_delay() (framework.monitor.Monitor method), 250

get_probe_delay() (framework.monitor.ProbeUser method), 257
get_probe_delay() (framework.plumbing.FmkPlumbing method), 299
get_probe_related_tg() (framework.monitor.Monitor method), 250
get_probe_status() (framework.monitor.Monitor method), 250
get_probe_status() (framework.monitor.ProbeUser method), 257
get_probes() (framework.project.Project method), 244
get_probes_names() (framework.monitor.Monitor method), 250
get_project_by_name() (framework.plumbing.FmkPlumbing method), 299
get_project_record() (framework.database.Database method), 274
get_projects() (framework.plumbing.FmkPlumbing method), 299
get_random_disruptor() (framework.tactics_helpers.Tactics method), 265
get_random_generator() (framework.tactics_helpers.Tactics method), 265
get_raw_value() (framework.node.NodeInternals method), 177
get_raw_value() (framework.node.NodeInternals_Empty method), 178
get_raw_value() (framework.node.NodeInternals_GenFunc method), 180
get_raw_value() (framework.node.NodeInternals_NonTerm method), 183
get_raw_value() (framework.node.NodeInternals_Term method), 185
get_raw_value() (framework.node.NodeInternals_TypedValue method), 186
get_reachable_nodes() (framework.node.Node method), 171
get_semantics() (framework.node.Node method), 172
get_separator_node() (framework.node.NodeInternals_NonTerm method), 183
get_solution() (framework.constraint_helpers.CSP method), 297
get_specific_fuzzy_vals() (framework.value_types.INT method), 206
get_specific_fuzzy_vals() (framework.value_types.VT method), 218
get_specific_fuzzy_vals() (framework.value_types.VT_Alt method), 218
get_specific_fuzzy_values() (framework.node.NodeInternals_TypedValue method), 186
get_subfield() (framework.value_types.BitField method), 201
get_subnode() (framework.node.NodeInternals_NonTerm method), 183
get_subnode_default_qty() (framework.node.NodeInternals_NonTerm method), 183
get_subnode_idx() (framework.node.NodeInternals_NonTerm method), 183
get_subnode_minmax() (framework.node.NodeInternals_NonTerm method), 183
get_subnode_off() (framework.node.NodeInternals_NonTerm method), 183
get_subnode_qty() (framework.node.NodeInternals_NonTerm method), 183
get_subnodes_collection() (framework.node.NodeInternals_NonTerm method), 183
get_subnodes_csts_copy() (framework.node.NodeInternals_NonTerm method), 183
get_subnodes_with_csts() (framework.node.NodeInternals_NonTerm method), 183
get_target_ip() (framework.targets.printer.PrinterTarget method), 241
get_target_path() (framework.targets.local.LocalTarget method), 238
get_target_port() (framework.targets.printer.PrinterTarget method), 241
get_tasks_description() (framework.scenario.Step method), 280
get_tasks_ref() (framework.scenario.Step method), 280
get_timestamp() (framework.knowledge.feedback_collector.FeedbackCollector method), 291
get_timestamp() (framework.monitor.ProbeStatus method), 256
get_timestamp() (framework.operator_helpers.LastInstruction method), 245
get_value() (framework.node.Node method), 172

get_value() (*framework.value_types.BitField method*), 201
get_value() (*framework.value_types.INT method*), 206
get_value() (*framework.value_types.String method*), 214
get_value() (*framework.value_types.VT method*), 218
get_value_type() (*framework.node.NodeInternals_TypedValue method*), 186
graph() (*framework.scenario.Scenario method*), 278
graph_info (*framework.node.DynNode_Helpers property*), 164
graph_scenario() (*framework.tactics_helpers.DynGeneratorFromScenario method*), 263
GSM7bitPacking (*class in framework.value_types*), 204
GSM7bitPacking_Enc (*class in framework.encoders*), 271
GSMPhoneNum (*class in framework.value_types*), 205
GSMPhoneNum_Enc (*class in framework.encoders*), 272
GZIP (*class in framework.value_types*), 205
GZIP_Enc (*class in framework.encoders*), 272

H

handle_data_desc() (*framework.plumbing.FmkPlumbing method*), 299
HandOver (*framework.tactics_helpers.DataMakerAttr attribute*), 262
handover() (*framework.tactics_helpers.StatefulDisruptor method*), 264
Hardware (*class in framework.knowledge.information*), 294
has_callback() (*framework.scenario.Transition method*), 282
has_callback_pending() (*framework.scenario.Transition method*), 282
has_dataprocess() (*framework.scenario.Step method*), 280
has_fbk_collector() (*framework.knowledge.feedback_collector.FeedbackCollector method*), 291
has_info() (*framework.data.Data method*), 157
has_node_constraints() (*framework.node.NodeInternalsCriteria method*), 178
has_node_content() (*framework.data.Data method*), 157
has_raw_content() (*framework.data.Data method*), 157
has_subkinds() (*framework.node.NodeInternals method*), 177
has_subkinds() (*framework.node.NodeInternals_TypedValue method*), 186
has_tasks_to_start() (*framework.scenario.Step method*), 281
has_tasks_to_stop() (*framework.scenario.Step method*), 281
HIGH_PRIO (*framework.node_builder.NodeBuilder attribute*), 190
Highlight (*framework.dmhelpers.generic.MH.Attr attribute*), 284
Highlight (*framework.node.NodeInternals attribute*), 176
highlight (*framework.node.NodeInternals property*), 177
highlight_variables (*framework.constraint_helpers.CSP attribute*), 297

I

ic (*framework.node_builder.NodeBuilder attribute*), 192
idx_from_desc() (*framework.value_types.BitField method*), 201
import_file_contents() (*framework.data_model.DataModel method*), 162
import_func() (*framework.node.NodeInternals_Func method*), 179
import_generator_func() (*framework.node.NodeInternals_GenFunc method*), 180
import_subnodes_basic() (*framework.node.NodeInternals_NonTerm method*), 183
import_subnodes_full_format() (*framework.node.NodeInternals_NonTerm method*), 183
import_subnodes_with_csts() (*framework.node.NodeInternals_NonTerm method*), 183
import_value_type() (*framework.node.NodeInternals_TypedValue method*), 186
included_models (*framework.data_model.DataModel property*), 162
IncorrectTargetError, 242
increase_trust() (*framework.knowledge.information.Info method*), 294
Individual (*class in framework.evolutionary_helpers*), 290
Inexistence (*framework.node.SyncScope attribute*), 189
INFINITY_LIMIT (*framework.node.NodeInternals_NonTerm attribute*), 180
Info (*class in framework.knowledge.information*), 294
info_exists() (*framework.data.Data method*), 157

InformationCollector (*class in framework.knowledge.information*), 295
init_command (*framework.monitor.ProbeCmd attribute*), 252, 253
init_encoder (*framework.value_types.String attribute*), 214
init_encoder() (*framework.value_types.GSM7bitPacking method*), 205
init_encoder() (*framework.value_types.GSMPhoneNum method*), 205
init_encoder() (*framework.value_types.GZIP method*), 205
init_encoder() (*framework.value_types.Wrapper method*), 219
init_encoding_scheme() (*framework.encoders.Encoder method*), 271
init_encoding_scheme() (*framework.encoders.GZIP_Enc method*), 273
init_encoding_scheme() (*framework.encoders.Wrap_Enc method*), 273
init_specific() (*framework.fuzzing_primitives.AltConfConsumer method*), 267
init_specific() (*framework.fuzzing_primitives.BasicVisitor method*), 267
init_specific() (*framework.fuzzing_primitives.NodeConsumerStub method*), 268
init_specific() (*framework.fuzzing_primitives.NonTermVisitor method*), 269
init_specific() (*framework.fuzzing_primitives.SeparatorDisruption method*), 270
init_specific() (*framework.fuzzing_primitives.TypedNodeDisruption method*), 270
init_specific() (*framework.node_builder.RegexParser method*), 199
init_specific() (*framework.node_builder.State method*), 199
INITIAL (*framework.node_builder.RegexParser.Brackets attribute*), 192
INITIAL (*framework.node_builder.RegexParser.Brackets.Comma attribute*), 192
INITIAL (*framework.node_builder.RegexParser.Brackets.Final attribute*), 192
INITIAL (*framework.node_builder.RegexParser.Brackets.Initial attribute*), 192
INITIAL (*framework.node_builder.RegexParser.Brackets.Max attribute*), 193
INITIAL (*framework.node_builder.RegexParser.Brackets.Min attribute*), 193
INITIAL (*framework.node_builder.RegexParser.Choice attribute*), 193
INITIAL (*framework.node_builder.RegexParser.Dot attribute*), 193
INITIAL (*framework.node_builder.RegexParser.Escape attribute*), 194
INITIAL (*framework.node_builder.RegexParser.EscapeMetaSequence attribute*), 194
INITIAL (*framework.node_builder.RegexParser.Final attribute*), 194
INITIAL (*framework.node_builder.RegexParser.Initial attribute*), 194
INITIAL (*framework.node_builder.RegexParser.Main attribute*), 195
INITIAL (*framework.node_builder.RegexParser.Parenthesis attribute*), 196
INITIAL (*framework.node_builder.RegexParser.Parenthesis.Choice attribute*), 195
INITIAL (*framework.node_builder.RegexParser.Parenthesis.Escape attribute*), 195
INITIAL (*framework.node_builder.RegexParser.Parenthesis.Final attribute*), 195
INITIAL (*framework.node_builder.RegexParser.Parenthesis.Initial attribute*), 196
INITIAL (*framework.node_builder.RegexParser.Parenthesis.Main attribute*), 196
INITIAL (*framework.node_builder.RegexParser.QtyState attribute*), 196
INITIAL (*framework.node_builder.RegexParser.SquareBrackets attribute*), 198
INITIAL (*framework.node_builder.RegexParser.SquareBrackets.AfterRange attribute*), 197
INITIAL (*framework.node_builder.RegexParser.SquareBrackets.BeforeRan attribute*), 197
INITIAL (*framework.node_builder.RegexParser.SquareBrackets.EscapeAfte attribute*), 197
INITIAL (*framework.node_builder.RegexParser.SquareBrackets.EscapeBef attribute*), 197
INITIAL (*framework.node_builder.RegexParser.SquareBrackets.EscapeMet attribute*), 198
INITIAL (*framework.node_builder.RegexParser.SquareBrackets.Final attribute*), 198
INITIAL (*framework.node_builder.RegexParser.SquareBrackets.Initial attribute*), 198
INITIAL (*framework.node_builder.RegexParser.SquareBrackets.Range attribute*), 198
initial() (*in module framework.node_builder*), 200
initialize() (*framework.targets.local.LocalTarget method*), 238
initialize() (*framework.targets.network.NetworkTarget method*),

236
input (*framework.node_builder.StateMachine property*), 200
InputHandling (*class in framework.knowledge.information*), 295
insert_analysis() (*framework.database.Database method*), 274
insert_async_data() (*framework.database.Database method*), 274
insert_comment() (*framework.database.Database method*), 274
insert_data() (*framework.database.Database method*), 274
insert_data_model() (*framework.database.Database method*), 275
insert_dmaker() (*framework.database.Database method*), 275
insert_feedback() (*framework.database.Database method*), 275
insert_fmk_info() (*framework.database.Database method*), 275
insert_project() (*framework.database.Database method*), 275
insert_steps() (*framework.database.Database method*), 275
INT (*class in framework.value_types*), 205
INT16 (*class in framework.value_types*), 207
INT32 (*class in framework.value_types*), 207
INT64 (*class in framework.value_types*), 207
INT8 (*class in framework.value_types*), 208
INT_str (*class in framework.value_types*), 208
IntCondition (*class in framework.node*), 166
interested_by() (*framework.fuzzing_primitives.NodeConsumerStub method*), 268
internals (*framework.node.Node attribute*), 166
invert_conditions() (*framework.scenario.Transition method*), 282
is_alive() (*framework.monitor.ProbeUser method*), 257
is_assumption_valid() (*framework.knowledge.information.InformationCollector method*), 295
is_attr_set() (*framework.node.Node method*), 172
is_attr_set() (*framework.node.NodeInternals method*), 177
is_attr_set() (*framework.tactics_helpers.Disruptor method*), 262
is_attr_set() (*framework.tactics_helpers.Generator method*), 264
is_attr_set() (*framework.tactics_helpers.StatefulDisruptor method*), 264
is_blocked() (*framework.data.Data method*), 157
is_blocked() (*framework.scenario.Step method*), 281
is_compatible() (*framework.value_types.BitField method*), 201
is_compatible() (*framework.value_types.INT method*), 206
is_compatible() (*framework.value_types.INT_str method*), 208
is_conf_existing() (*framework.node.Node method*), 172
is_crossable() (*framework.scenario.Transition method*), 282
is_current_solution_queried (*framework.constraint_helpers.CSP property*), 297
is_djob_registered() (*framework.node.Env method*), 164
is_empty() (*framework.data.Data method*), 157
is_empty() (*framework.node.Env method*), 164
is_empty() (*framework.node.Env4NT method*), 165
is_empty() (*framework.node.Node method*), 172
is_enabled (*libs.utils.ExternalDisplay property*), 302
is_enabled() (*framework.database.Database method*), 275
is_exhausted() (*framework.node.Node method*), 172
is_exhausted() (*framework.node.NodeInternals method*), 177
is_exhausted() (*framework.node.NodeInternals_GenFunc method*), 180
is_exhausted() (*framework.node.NodeInternals_NonTerm method*), 184
is_exhausted() (*framework.node.NodeInternals_Term method*), 185
is_exhausted() (*framework.node.NodeInternals_TypedValue method*), 186
is_exhausted() (*framework.value_types.BitField method*), 201
is_exhausted() (*framework.value_types.INT method*), 206
is_exhausted() (*framework.value_types.String method*), 214
is_exhausted() (*framework.value_types.VT method*), 218
is_feedback_received() (*framework.target_helpers.Target method*), 231
is_feedback_received() (*framework.targets.debug.TestTarget method*), 242
is_feedback_received() (*framework.targets.network.NetworkTarget method*), 236
is_feedback_received() (*framework.targets.ssh.SSHTarget method*), 240

is_final() (*framework.evolutionary_helpers.DefaultPopulation method*), 290
is_final() (*framework.evolutionary_helpers.Population method*), 291
is_flag_set() (*framework.data.CallBackOps method*), 156
is_flag_set() (*framework.operator_helpers.Operation method*), 246
is_frozen() (*framework.node.Node method*), 172
is_frozen() (*framework.node.NodeInternals method*), 177
is_frozen() (*framework.node.NodeInternals_GenFunc method*), 180
is_frozen() (*framework.node.NodeInternals_NonTerm method*), 184
is_frozen() (*framework.node.NodeInternals_Term method*), 185
is_func() (*framework.node.Node method*), 172
is_genfunc() (*framework.node.Node method*), 172
is_info_class_represented() (*framework.knowledge.information.InformationCollector method*), 295
is_instruction_set() (*framework.operator_helpers.LastInstruction method*), 245
is_node_exhausted() (*framework.node.Env method*), 165
is_nonterm() (*framework.node.Node method*), 172
is_not_ok() (*framework.plumbing.FmkPlumbing method*), 299
is_ok() (*framework.plumbing.FmkPlumbing method*), 299
is_path_valid() (*framework.node.Node method*), 172
is_periodic_cleared() (*framework.scenario.Step method*), 281
is_periodic_set() (*framework.scenario.Step method*), 281
is_probe_launched() (*framework.monitor.Monitor method*), 250
is_probe_stuck() (*framework.monitor.Monitor method*), 250
is_processed_data_altered() (*framework.target_helpers.Target method*), 231
is_recordable() (*framework.data.Data method*), 157
is_set() (*framework.data.AttrGroup method*), 155
is_size_compatible() (*framework.value_types.INT method*), 206
is_started() (*framework.target_helpers.Target method*), 231
is_stuck() (*framework.monitor.ProbeUser method*), 257
is_target_enabled() (*framework.plumbing.FmkPlumbing method*), 299
is_target_ok() (*framework.monitor.Monitor method*), 251
is_target_ready_for_new_data() (*framework.target_helpers.Target method*), 231
is_target_ready_for_new_data() (*framework.targets.debug.TestTarget method*), 242
is_target_ready_for_new_data() (*framework.targets.network.NetworkTarget method*), 236
is_term() (*framework.node.Node method*), 172
is_terminal (*libs.utils.ExternalDisplay property*), 302
is_typed_value() (*framework.node.Node method*), 172
is_unusable() (*framework.data.Data method*), 157
is_usable() (*framework.plumbing.FmkPlumbing method*), 299
iter_and_cleanup_collector() (*framework.knowledge.feedback_collector.FeedbackCollector method*), 291
iter_data_bank() (*framework.plumbing.FmkPlumbing method*), 299
iter_data_models() (*framework.plumbing.FmkPlumbing method*), 299
iter_entries() (*framework.database.FeedbackGate method*), 275
iter_feedback_entries() (*framework.database.Database method*), 275
iter_nodes_by_path() (*framework.node.Node method*), 172
iter_paths() (*framework.node.Node method*), 173
iter_probes() (*framework.monitor.Monitor method*), 251
iter_vars() (*framework.constraint_helpers.CSP method*), 297

J

join() (*framework.monitor.ProbeUser method*), 257

K

knowledge_source (*framework.data_model.DataModel attribute*), 162
knowledge_source (*framework.node.Env attribute*), 165
knowledge_source (*framework.project.Project property*), 244
knowledge_source (*framework.scenario.ScenarioEnv attribute*), 279
knowledge_source (*framework.tactics_helpers.DataMaker attribute*), 261
knowledge_source (*framework.value_types.VT attribute*), 218

L

Language (*class in framework.knowledge.information*), 295
LastInstruction (*class in framework.operator_helpers*), 245
LATIN_1 (*framework.value_types.String attribute*), 211
launch() (*framework.plumbing.FmkPlumbing method*), 299
launch_operator() (*framework.plumbing.FmkPlumbing method*), 299
launch_probe() (*framework.plumbing.FmkPlumbing method*), 299
Leaf (*framework.dmhelpers.generic.MH attribute*), 285
LEN() (*in module framework.dmhelpers.generic*), 284
libs.utils
    module, 302
Linux (*framework.knowledge.information.OS attribute*), 295
linux_prefix (*framework.value_types.Filename attribute*), 202
linux_specific_fnames (*framework.value_types.Filename attribute*), 202
linux_suffix (*framework.value_types.Filename attribute*), 203
listen_to() (*framework.targets.network.NetworkTarget method*), 236
LittleEndian (*framework.value_types.VT attribute*), 217
load_data_model() (*framework.data_model.DataModel method*), 162
load_data_model() (*framework.plumbing.FmkPlumbing method*), 299
load_multiple_data_model() (*framework.plumbing.FmkPlumbing method*), 299
load_project() (*framework.plumbing.FmkPlumbing method*), 299
load_targets() (*framework.plumbing.FmkPlumbing method*), 299
LocalTarget (*class in framework.targets.local*), 238
LOCKED (*framework.dmhelpers.generic.MH.Attr attribute*), 284
LOCKED (*framework.node.NodeInternals attribute*), 176
log_async_data() (*framework.logger.Logger method*), 248
log_collected_feedback() (*framework.logger.Logger method*), 248
log_comment() (*framework.logger.Logger method*), 248
log_comment() (*framework.plumbing.FmkPlumbing method*), 299
log_data() (*framework.logger.Logger method*), 248
log_data_info() (*framework.logger.Logger method*), 248
log_disruptor_info() (*framework.logger.Logger method*), 248
log_dmaker_step() (*framework.logger.Logger method*), 248
log_error() (*framework.logger.Logger method*), 248
log_fmk_info() (*framework.logger.Logger method*), 248
log_generator_info() (*framework.logger.Logger method*), 248
log_info() (*framework.logger.Logger method*), 249
log_operator_feedback() (*framework.logger.Logger method*), 249
log_probe_feedback() (*framework.logger.Logger method*), 249
log_target_ack_date() (*framework.logger.Logger method*), 249
log_target_feedback_from() (*framework.logger.Logger method*), 249
log_target_residual_feedback() (*framework.plumbing.FmkPlumbing method*), 299
Logger (*class in framework.logger*), 247
LOW_PRIO (*framework.node_builder.NodeBuilder attribute*), 190

M

main() (*framework.monitor.Probe method*), 252
main() (*framework.monitor.ProbeCmd method*), 253
main() (*framework.monitor.ProbeMem method*), 254
main() (*framework.monitor.ProbePID method*), 255
make_args_private() (*framework.node.NodeInternals_Func method*), 179
make_args_private() (*framework.node.NodeInternals_GenFunc method*), 180
make_blocked() (*framework.data.Data method*), 157
make_blocked() (*framework.data.DataProcess method*), 159
make_blocked() (*framework.scenario.Step method*), 281
make_determinist() (*framework.node.Node method*), 173
make_determinist() (*framework.value_types.BitField method*), 201
make_determinist() (*framework.value_types.INT method*), 206
make_determinist() (*framework.value_types.String method*), 214
make_determinist() (*framework.value_types.VT method*), 218
make_empty() (*framework.node.Node method*), 173
make_entangled_nodes() (*in module framework.node*), 190
make_finite() (*framework.node.Node method*), 173
make_free() (*framework.data.Data method*), 157

make_free() (*framework.data.DataProcess method*), 159
make_free() (*framework.scenario.NoDataStep method*), 277
make_free() (*framework.scenario.Step method*), 281
make_infinite() (*framework.node.Node method*), 173
make_private (*framework.node.NodeSeparator attribute*), 187
make_private() (*framework.node.DynNode_Helpers method*), 164
make_private() (*framework.node.NodeAbstraction method*), 175
make_private() (*framework.node.NodeInternals method*), 177
make_private() (*framework.node.NodeSemantics method*), 186
make_private() (*framework.node.NodeSeparator method*), 187
make_private() (*framework.node.SyncObj method*), 189
make_private() (*framework.value_types.BitField method*), 201
make_private() (*framework.value_types.INT method*), 206
make_private() (*framework.value_types.String method*), 214
make_private() (*framework.value_types.VT method*), 218
make_private_subnodes() (*framework.node.NodeInternals_NonTerm method*), 184
make_random() (*framework.node.Node method*), 173
make_random() (*framework.value_types.BitField method*), 201
make_random() (*framework.value_types.INT method*), 206
make_random() (*framework.value_types.String method*), 214
make_random() (*framework.value_types.VT method*), 218
make_recordable() (*framework.data.Data method*), 157
make_stutter() (*framework.scenario.Step method*), 281
make_synchronized_with() (*framework.node.Node method*), 173
make_uncrossable() (*framework.scenario.Transition method*), 282
make_unusable() (*framework.data.Data method*), 157
make_wrapped_node() (*in module framework.node*), 190
map_targets_to_scenario() (*framework.project.Project method*), 244
map_var_to_node() (*framework.constraint_helpers.CSP method*), 297
match() (*framework.node.NodeInternals method*), 177
match() (*framework.node.NodeSemantics method*), 187
max_attempts (*framework.monitor.ProbePID attribute*), 255
max_consumer (*framework.targets.debug.TestTarget attribute*), 243
max_nb_runs_for() (*framework.fuzzing_primitives.NodeConsumerStub method*), 268
maxi (*framework.value_types.INT attribute*), 206
maxi (*framework.value_types.SINT16_be attribute*), 209
maxi (*framework.value_types.SINT16_le attribute*), 209
maxi (*framework.value_types.SINT32_be attribute*), 209
maxi (*framework.value_types.SINT32_le attribute*), 210
maxi (*framework.value_types.SINT64_be attribute*), 210
maxi (*framework.value_types.SINT64_le attribute*), 210
maxi (*framework.value_types.SINT8 attribute*), 211
maxi (*framework.value_types.UINT16_be attribute*), 215
maxi (*framework.value_types.UINT16_le attribute*), 215
maxi (*framework.value_types.UINT32_be attribute*), 216
maxi (*framework.value_types.UINT32_le attribute*), 216
maxi (*framework.value_types.UINT64_be attribute*), 216
maxi (*framework.value_types.UINT64_le attribute*), 217
maxi (*framework.value_types.UINT8 attribute*), 217
maxi (*framework.value_types.VT attribute*), 218
maxi_gen (*framework.value_types.INT attribute*), 206
Maximum (*framework.knowledge.information.TrustLevel attribute*), 296
Medium (*framework.knowledge.information.Test attribute*), 296
Medium (*framework.knowledge.information.TrustLevel attribute*), 296
MEDIUM_PRIO (*framework.node_builder.NodeBuilder attribute*), 190
merge_user_context_with() (*framework.scenario.Scenario method*), 278
merge_with() (*framework.data_model.DataModel method*), 162
merge_with() (*framework.data_model.NodeBackend method*), 163
meta_data_size (*framework.targets.debug.TestTarget attribute*), 243
MH (*class in framework.dmhelpers.generic*), 284
MH.Attr (*class in framework.dmhelpers.generic*), 284
MH.Charset (*class in framework.dmhelpers.generic*), 285
MH.Custo (*class in framework.dmhelpers.generic*), 285
MH.Custo.Func (*class in framework.dmhelpers.generic*), 285
MH.Custo.Gen (*class in framework.dmhelpers.generic*), 285
MH.Custo.NTerm (*class in framework.dmhelpers.generic*), 285

mini (*framework.value_types.INT attribute*), 207
mini (*framework.value_types.SINT16_be attribute*), 209
mini (*framework.value_types.SINT16_le attribute*), 209
mini (*framework.value_types.SINT32_be attribute*), 209
mini (*framework.value_types.SINT32_le attribute*), 210
mini (*framework.value_types.SINT64_be attribute*), 210
mini (*framework.value_types.SINT64_le attribute*), 210
mini (*framework.value_types.SINT8 attribute*), 211
mini (*framework.value_types.UINT16_be attribute*), 215
mini (*framework.value_types.UINT16_le attribute*), 216
mini (*framework.value_types.UINT32_be attribute*), 216
mini (*framework.value_types.UINT32_le attribute*), 216
mini (*framework.value_types.UINT64_be attribute*), 216
mini (*framework.value_types.UINT64_le attribute*), 217
mini (*framework.value_types.UINT8 attribute*), 217
mini (*framework.value_types.VT attribute*), 218
mini_gen (*framework.value_types.INT attribute*), 207
Minimum (*framework.knowledge.information.TrustLevel attribute*), 296
ModelWalker (*class in framework.fuzzing_primitives*), 267
modelwalker_inputs_handling_helper() (*in module framework.tactics_helpers*), 266
modelwalker_user (*framework.tactics_helpers.DataMaker property*), 261
module
    framework.basic_primitives, 155
    framework.comm_backends, 257
    framework.constraint_helpers, 296
    framework.data, 155
    framework.data_model, 160
    framework.database, 273
    framework.dmhelpers.generic, 283
    framework.dmhelpers.xml, 288
    framework.encoders, 271
    framework.evolutionary_helpers, 289
    framework.fuzzing_primitives, 266
    framework.generic_data_makers, 219
    framework.knowledge.feedback_collector, 291
    framework.knowledge.feedback_handler, 292
    framework.knowledge.information, 294
    framework.logger, 247
    framework.monitor, 249
    framework.node, 163
    framework.node_builder, 190
    framework.operator_helpers, 245
    framework.plumbing, 298
    framework.project, 244
    framework.scenario, 276
    framework.tactics_helpers, 261
    framework.target_helpers, 229
    framework.targets.debug, 242
    framework.targets.local, 238
    framework.targets.network, 233
    framework.targets.printer, 241
    framework.targets.ssh, 239
    framework.value_types, 200
    libs.utils, 302
Monitor (*class in framework.monitor*), 250
monitor_probes() (*framework.plumbing.FmkPlumbing method*), 299
Mutable (*framework.dmhelpers.generic.MH.Attr attribute*), 284
Mutable (*framework.node.NodeInternals attribute*), 176
mutable_clone_mode (*framework.node.NonTermCusto property*), 188
MutableClone (*framework.dmhelpers.generic.MH.Custo.NTerm attribute*), 285
MutableClone (*framework.node.NonTermCusto attribute*), 188
mutate() (*framework.evolutionary_helpers.DefaultIndividual method*), 289
mutate() (*framework.evolutionary_helpers.Individual method*), 290

N

name (*framework.data_model.DataModel attribute*), 162
name (*framework.node.Node attribute*), 166
name (*framework.project.Project attribute*), 245
name (*framework.target_helpers.Target attribute*), 231
Native (*framework.value_types.VT attribute*), 217
nb_constraints (*framework.constraint_helpers.CSP property*), 297
need_reset() (*framework.fuzzing_primitives.AltConfConsumer method*), 267
need_reset() (*framework.fuzzing_primitives.BasicVisitor method*), 267
need_reset() (*framework.fuzzing_primitives.NodeConsumerStub method*), 268
need_reset() (*framework.fuzzing_primitives.NonTermVisitor method*), 269
need_reset() (*framework.fuzzing_primitives.TypedNodeDisruption method*), 270
need_reset() (*framework.tactics_helpers.Generator method*), 264
NeedSeed (*framework.tactics_helpers.DataMakerAttr attribute*), 262
negate() (*framework.constraint_helpers.Constraint method*), 297

negate_constraint() (*framework.constraint_helpers.CSP method*), 297
NetworkTarget (*class in framework.targets.network*), 233
next() (*framework.evolutionary_helpers.Population method*), 291
next_process() (*framework.data.DataProcess method*), 159
next_qty() (*framework.node.NodeInternals_NonTerm.NodeAttrs method*), 181
next_solution() (*framework.constraint_helpers.CSP method*), 297
no_more_solution_for_csp (*framework.node.Node property*), 173
NO_PASSWORD (*framework.comm_backends.SSH_Backend attribute*), 258
NO_PASSWORD (*framework.targets.ssh.SSHTarget attribute*), 239
NoDataStep (*class in framework.scenario*), 276
Node (*class in framework.node*), 166
node_consumer_helper() (*framework.fuzzing_primitives.ModelWalker method*), 268
node_exists() (*framework.node.Env4NT method*), 165
NodeAbstraction (*class in framework.node*), 175
NodeBackend (*class in framework.data*), 159
NodeBackend (*class in framework.data_model*), 162
NodeBuilder (*class in framework.node_builder*), 190
NodeCondition (*class in framework.node*), 175
NodeConsumerStub (*class in framework.fuzzing_primitives*), 268
NodeCustomization (*class in framework.node*), 175
NodeInternals (*class in framework.node*), 176
NodeInternals_Empty (*class in framework.node*), 178
NodeInternals_Func (*class in framework.node*), 178
NodeInternals_GenFunc (*class in framework.node*), 179
NodeInternals_NonTerm (*class in framework.node*), 180
NodeInternals_NonTerm.NodeAttrs (*class in framework.node*), 180
NodeInternals_Term (*class in framework.node*), 184
NodeInternals_TypedValue (*class in framework.node*), 185
NodeInternalsCriteria (*class in framework.node*), 178
nodeqty_corrupt_hook() (*framework.node.NodeInternals_NonTerm static method*), 184
NodeSemantics (*class in framework.node*), 186
NodeSemanticsCriteria (*class in framework.node*), 187
NodeSeparator (*class in framework.node*), 187
non_ctrl_char (*framework.value_types.String attribute*), 214
NonTermCusto (*class in framework.node*), 188
NonTerminal (*framework.dmhelpers.generic.MH attribute*), 285
NonTermVisitor (*class in framework.fuzzing_primitives*), 269
notify_blocking() (*framework.monitor.BlockingProbeUser method*), 250
notify_data_ready() (*framework.monitor.BlockingProbeUser method*), 250
notify_data_sending() (*framework.knowledge.feedback_handler.FeedbackHandler method*), 293
notify_data_sending() (*framework.project.Project method*), 245
notify_data_sending_event() (*framework.monitor.Monitor method*), 251
notify_error() (*framework.monitor.BlockingProbeUser method*), 250
notify_error() (*framework.monitor.Monitor method*), 251
notify_exhausted_node() (*framework.node.Env method*), 165
notify_imminent_data_sending() (*framework.monitor.Monitor method*), 251
notify_target_feedback_retrieval() (*framework.monitor.Monitor method*), 251

O

obj (*framework.knowledge.feedback_collector.FeedbackSource property*), 292
OFFSET() (*in module framework.dmhelpers.generic*), 286
Operation (*class in framework.operator_helpers*), 246
OperationMode (*class in framework.knowledge.information*), 295
Operator (*class in framework.operator_helpers*), 246
operator() (*in module framework.operator_helpers*), 246
Ordered (*framework.dmhelpers.generic.MH attribute*), 285
origin (*framework.data.Data property*), 157
OS (*class in framework.knowledge.information*), 295
OUTCOME_DATA (*framework.database.Database attribute*), 273
OUTCOME_ROWID (*framework.database.Database attribute*), 273

P

padding_one (*framework.value_types.BitField attribute*), 201

parse() (*framework.node_builder.RegexParser method*), 199
Pascal (*framework.knowledge.information.Language attribute*), 295
path_mode (*framework.value_types.Filename attribute*), 203
pending_callback_ops() (*framework.data.Data method*), 157
perform_planned_reset() (*framework.node.NodeInternals_NonTerm.NodeAttrs method*), 181
period (*libs.utils.Task attribute*), 302
Periodic (*class in framework.scenario*), 277
periodic_to_clear (*framework.scenario.Scenario property*), 278
periodic_to_clear (*framework.scenario.Step property*), 281
periodic_to_set (*framework.scenario.Step property*), 281
Pick (*framework.dmhelpers.generic.MH attribute*), 285
plan_next_operation() (*framework.operator_helpers.Operator method*), 246
plan_reset() (*framework.node.NodeInternals_NonTerm.NodeAttrs method*), 181
Population (*class in framework.evolutionary_helpers*), 290
PowerPc (*framework.knowledge.information.Hardware attribute*), 294
pre_build() (*framework.data_model.DataModel method*), 162
preload() (*framework.fuzzing_primitives.NodeConsumerStub method*), 268
preload() (*framework.fuzzing_primitives.TypedNodeDisruption method*), 270
prepare_atom() (*framework.data_model.NodeBackend method*), 163
pretty_print() (*framework.data.Data method*), 157
pretty_print() (*framework.node.Node method*), 173
pretty_print() (*framework.node.NodeInternals method*), 177
pretty_print() (*framework.node.NodeInternals_TypedValue method*), 186
pretty_print() (*framework.value_types.BitField method*), 201
pretty_print() (*framework.value_types.INT method*), 207
pretty_print() (*framework.value_types.INT_str method*), 208
pretty_print() (*framework.value_types.String method*), 214
pretty_print() (*framework.value_types.VT method*), 218
PRETTY_PRINT_API (*framework.logger.Logger attribute*), 247
pretty_print_data() (*framework.logger.Logger method*), 249
print() (*framework.knowledge.feedback_handler.FeedbackHandler method*), 293
print() (*framework.plumbing.Printer method*), 301
print() (*libs.utils.Task method*), 302
print() (*libs.utils.Term method*), 302
print_console() (*framework.logger.Logger method*), 249
PRINT_CONSOLE_API (*framework.logger.Logger attribute*), 247
print_disruptor() (*framework.tactics_helpers.Tactics method*), 265
print_generator() (*framework.tactics_helpers.Tactics method*), 265
print_nl() (*framework.knowledge.feedback_handler.FeedbackHandler method*), 293
print_nl() (*libs.utils.Task method*), 302
print_nl() (*libs.utils.Term method*), 302
Printable_Char_Set (*framework.knowledge.information.InputHandling attribute*), 295
printable_char_set (*framework.value_types.String attribute*), 214
Printer (*class in framework.plumbing*), 301
PrinterTarget (*class in framework.targets.printer*), 241
prj (*framework.plumbing.FmkPlumbing property*), 299
prj (*libs.utils.Task attribute*), 302
Probe (*class in framework.monitor*), 251
probe (*framework.monitor.ProbeUser property*), 257
probe() (*in module framework.monitor*), 257
probe_init_timeout (*framework.monitor.ProbeUser attribute*), 257
probe_name (*framework.monitor.AddExistingProbeToMonitorError property*), 249
probe_name (*framework.monitor.ProbeTimeoutError property*), 256
ProbeCmd (*class in framework.monitor*), 252
ProbeMem (*class in framework.monitor*), 253
ProbePID (*class in framework.monitor*), 254
ProbeStatus (*class in framework.monitor*), 256
ProbeTimeoutError, 256
ProbeUser (*class in framework.monitor*), 256
proc_instr (*framework.dmhelpers.xml.TAG_TYPE attribute*), 288
process (*framework.data.DataProcess property*), 159
process_data() (*framework.plumbing.FmkPlumbing method*), 299
process_data_and_send() (*framework.plumbing.FmkPlumbing method*), 299

process_feedback() (*framework.knowledge.feedback_handler.FeedbackHandler method*), 293
process_name (*framework.monitor.ProbeMem attribute*), 253, 254
process_name (*framework.monitor.ProbePID attribute*), 255, 256
process_qty (*framework.data.DataProcess property*), 159
produced_seed (*framework.tactics_helpers.DynGeneratorFromScenario property*), 263
produced_seed (*framework.tactics_helpers.Generator attribute*), 264
producer_status_idx (*framework.targets.debug.TestTarget attribute*), 243
Project (*class in framework.project*), 244
projects() (*framework.plumbing.FmkPlumbing method*), 300
put_node_containers() (*framework.node.SyncExistenceObj method*), 189
put_node_containers() (*framework.node.SyncObj method*), 189
put_node_containers() (*framework.node.SyncQtyFromObj method*), 189
put_node_containers() (*framework.node.SyncSizeObj method*), 190

Q

qty (*framework.node.NodeInternals_NonTerm.NodeAttrs property*), 181
qty (*framework.node.SyncQtyFromObj property*), 189
Qty (*framework.node.SyncScope attribute*), 189
QTY() (*in module framework.dmhelpers.generic*), 286
qty_sequence (*framework.node.NodeInternals_NonTerm.NodeAttrs property*), 181
QtyFrom (*framework.node.SyncScope attribute*), 190
qtysync_corrupt_hook() (*framework.node.NodeInternals_NonTerm static method*), 184

R

rand_string() (*in module framework.basic_primitives*), 155
Random (*framework.dmhelpers.generic.MH attribute*), 286
Random (*framework.knowledge.information.OperationMode attribute*), 296
RawBackend (*class in framework.data*), 160
RawCondition (*class in framework.node*), 188
RawNode (*framework.dmhelpers.generic.MH attribute*), 286
read_info() (*framework.data.Data method*), 157
read_output() (*framework.comm_backends.Backend method*), 257
read_output() (*framework.comm_backends.Serial_Backend method*), 260
read_output() (*framework.comm_backends.Shell_Backend method*), 261
read_output() (*framework.comm_backends.SSH_Backend method*), 259
read_stderr() (*framework.comm_backends.SSH_Backend method*), 259
read_stdout() (*framework.comm_backends.SSH_Backend method*), 259
record_info() (*framework.target_helpers.Target method*), 231
RecordData (*framework.operator_helpers.LastInstruction attribute*), 245
recover_node() (*framework.fuzzing_primitives.AltConfConsumer method*), 267
recover_node() (*framework.fuzzing_primitives.BasicVisitor method*), 267
recover_node() (*framework.fuzzing_primitives.NodeConsumerStub method*), 269
recover_node() (*framework.fuzzing_primitives.TypedNodeDisruption method*), 270
recover_target() (*framework.target_helpers.Target method*), 232
recover_target() (*framework.targets.debug.TestTarget method*), 243
recover_target() (*framework.targets.network.NetworkTarget method*), 237
recover_target() (*framework.targets.ssh.SSHTarget method*), 240
recurrent_command (*framework.monitor.ProbeCmd attribute*), 252, 253
Regex (*framework.dmhelpers.generic.MH attribute*), 286
regex_bin (*framework.value_types.INT_str attribute*), 208
regex_decimal (*framework.value_types.INT_str attribute*), 208
regex_lower_hex (*framework.value_types.INT_str attribute*), 208
regex_octal (*framework.value_types.INT_str attribute*), 208

regex_upper_hex (*framework.value_types.INT_str attribute*), 208
regexp() (*in module framework.database*), 276
regexp_bin() (*in module framework.database*), 276
RegexParser (*class in framework.node_builder*), 192
RegexParser.Brackets (*class in framework.node_builder*), 192
RegexParser.Brackets.Comma (*class in framework.node_builder*), 192
RegexParser.Brackets.Final (*class in framework.node_builder*), 192
RegexParser.Brackets.Initial (*class in framework.node_builder*), 192
RegexParser.Brackets.Max (*class in framework.node_builder*), 193
RegexParser.Brackets.Min (*class in framework.node_builder*), 193
RegexParser.Choice (*class in framework.node_builder*), 193
RegexParser.Dot (*class in framework.node_builder*), 193
RegexParser.Escape (*class in framework.node_builder*), 194
RegexParser.EscapeMetaSequence (*class in framework.node_builder*), 194
RegexParser.Final (*class in framework.node_builder*), 194
RegexParser.Group (*class in framework.node_builder*), 194
RegexParser.Initial (*class in framework.node_builder*), 194
RegexParser.Main (*class in framework.node_builder*), 195
RegexParser.Parenthesis (*class in framework.node_builder*), 195
RegexParser.Parenthesis.Choice (*class in framework.node_builder*), 195
RegexParser.Parenthesis.Escape (*class in framework.node_builder*), 195
RegexParser.Parenthesis.Final (*class in framework.node_builder*), 195
RegexParser.Parenthesis.Initial (*class in framework.node_builder*), 196
RegexParser.Parenthesis.Main (*class in framework.node_builder*), 196
RegexParser.QtyState (*class in framework.node_builder*), 196
RegexParser.SquareBrackets (*class in framework.node_builder*), 196
RegexParser.SquareBrackets.AfterRange (*class in framework.node_builder*), 197
RegexParser.SquareBrackets.BeforeRange (*class in framework.node_builder*), 197
RegexParser.SquareBrackets.EscapeAfterRange (*class in framework.node_builder*), 197
RegexParser.SquareBrackets.EscapeBeforeRange (*class in framework.node_builder*), 197
RegexParser.SquareBrackets.EscapeMetaSequence (*class in framework.node_builder*), 198
RegexParser.SquareBrackets.Final (*class in framework.node_builder*), 198
RegexParser.SquareBrackets.Initial (*class in framework.node_builder*), 198
RegexParser.SquareBrackets.Range (*class in framework.node_builder*), 198
register() (*framework.data_model.DataModel method*), 162
register() (*in module framework.node_builder*), 200
register_atom_for_decoding() (*framework.data_model.DataModel method*), 162
register_basic_djob() (*framework.node.Env method*), 165
register_callback() (*framework.data.Data method*), 157
register_callback() (*framework.scenario.Transition method*), 282
register_current_in_data_bank() (*framework.plumbing.FmkPlumbing method*), 300
register_djob() (*framework.node.Env method*), 165
register_evolutionary_processes() (*framework.project.Project method*), 245
register_feedback_handler() (*framework.project.Project method*), 245
register_in_data_bank() (*framework.plumbing.FmkPlumbing method*), 300
register_last_in_data_bank() (*framework.plumbing.FmkPlumbing method*), 300
register_new_disruptor() (*framework.tactics_helpers.Tactics method*), 265
register_new_generator() (*framework.tactics_helpers.Tactics method*), 266
register_new_interface() (*framework.targets.network.NetworkTarget method*), 237
register_operator() (*framework.project.Project method*), 245
register_post_freeze_handler() (*framework.node.Node method*), 173
register_probe() (*framework.project.Project method*), 245
register_scenarios() (*framework.project.Project method*), 245
register_scenarios() (*framework.tactics_helpers.Tactics method*), 266
reinit_steps (*framework.scenario.Scenario property*), 278
reinit_transitions (*framework.scenario.Scenario property*), 278

related_dm_name (*framework.tactics_helpers.DataMaker attribute*), 261
related_tg (*framework.knowledge.feedback_collector.FeedbackSource property*), 292
relation (*framework.constraint_helpers.Constraint attribute*), 297
reload_all() (*framework.plumbing.FmkPlumbing method*), 300
reload_dm() (*framework.plumbing.FmkPlumbing method*), 300
remove_all_dynamic_interfaces() (*framework.targets.network.NetworkTarget method*), 237
remove_conf() (*framework.node.Node method*), 173
remove_data() (*framework.database.Database method*), 275
remove_djob() (*framework.node.Env method*), 165
remove_dynamic_interface() (*framework.targets.network.NetworkTarget method*), 237
remove_info_from() (*framework.data.Data method*), 157
remove_node_to_corrupt() (*framework.node.Env method*), 165
RemoveCB (*framework.data.CallBackOps attribute*), 156
Replace_Data (*framework.data.CallBackOps attribute*), 156
replace_subnode() (*framework.node.NodeInternals_NonTerm method*), 184
reset() (*framework.constraint_helpers.CSP method*), 297
reset() (*framework.data.DataProcess method*), 159
reset() (*framework.encoders.Encoder method*), 271
reset() (*framework.evolutionary_helpers.DefaultPopulation method*), 290
reset() (*framework.evolutionary_helpers.Population method*), 291
reset() (*framework.monitor.Probe method*), 252
reset() (*framework.monitor.ProbeMem method*), 254
reset() (*framework.node.Env4NT method*), 165
reset() (*framework.node.NodeInternals_NonTerm method*), 184
reset() (*framework.node.NodeInternals_NonTerm.NodeAttr method*), 181
reset() (*framework.node_builder.RegexParser method*), 199
reset() (*framework.scenario.Scenario method*), 278
reset_constraint() (*framework.constraint_helpers.CSP method*), 297
reset_current_state() (*framework.logger.Logger method*), 249
reset_depth_specific() (*framework.node.NodeInternals method*), 177
reset_depth_specific() (*framework.node.NodeInternals_GenFunc method*), 180
reset_depth_specific() (*framework.node.NodeInternals_NonTerm method*), 184
reset_depth_specific() (*framework.node.NodeInternals_Term method*), 185
Reset_DMakers (*framework.data.DataAttr attribute*), 158
reset_encoder() (*framework.value_types.String method*), 214
reset_fuzz_weight() (*framework.node.Node method*), 173
reset_fuzz_weight() (*framework.node.NodeInternals_GenFunc method*), 180
reset_fuzz_weight() (*framework.node.NodeInternals_NonTerm method*), 184
reset_fuzz_weight() (*framework.node.NodeInternals_Term method*), 185
reset_generator() (*framework.node.NodeInternals_GenFunc method*), 180
reset_graph_info() (*framework.node.DynNode_Helpers method*), 164
reset_history() (*framework.data.Data method*), 157
reset_information() (*framework.knowledge.information.InformationCollector method*), 295
reset_knowledge() (*framework.project.Project method*), 245
reset_on_unfreeze_mode (*framework.node.GenFuncCusto property*), 166
reset_state() (*framework.fuzzing_primitives.BasicVisitor method*), 267
reset_state() (*framework.fuzzing_primitives.NodeConsumerStub method*), 269
reset_state() (*framework.node.Node method*), 173
reset_state() (*framework.node.NodeInternals_GenFunc method*), 180
reset_state() (*framework.node.NodeInternals_NonTerm method*), 184
reset_state() (*framework.node.NodeInternals_Term method*), 185
reset_state() (*framework.value_types.BitField*

method), 201
reset_state() (*framework.value_types.INT method*), 207
reset_state() (*framework.value_types.String method*), 214
reset_state() (*framework.value_types.VT method*), 218
reset_target_mappings() (*framework.project.Project method*), 245
reset_to_original() (*framework.constraint_helpers.Constraint method*), 297
reset_trust() (*framework.knowledge.information.Info method*), 294
ResetOnUnfreeze (*framework.dmhelpers.generic.MH.Custo.Gen attribute*), 285
ResetOnUnfreeze (*framework.node.GenFuncCusto attribute*), 166
restore_var_domains() (*framework.constraint_helpers.CSP method*), 297
retrieve_and_log_target_feedback() (*framework.plumbing.FmkPlumbing method*), 300
retrieve_app_handler() (*in module libs.utils*), 303
rewind() (*framework.value_types.BitField method*), 201
rewind() (*framework.value_types.INT method*), 207
rewind() (*framework.value_types.String method*), 214
rewind() (*framework.value_types.VT method*), 218
RootNS (*framework.node_builder.NodeBuilder attribute*), 190
run() (*framework.node_builder.State method*), 199
run() (*framework.node_builder.StateMachine method*), 200
run() (*framework.plumbing.FmkTask method*), 301
run_callback() (*framework.scenario.Transition method*), 282
run_callbacks() (*framework.data.Data method*), 157
run_project() (*framework.plumbing.FmkPlumbing method*), 300

S

save_current_var_domains() (*framework.constraint_helpers.CSP method*), 297
save_node() (*framework.fuzzing_primitives.AltConfConsumer method*), 267
save_node() (*framework.fuzzing_primitives.BasicVisitor method*), 267
save_node() (*framework.fuzzing_primitives.NodeConsumerStub method*), 269
save_node() (*framework.fuzzing_primitives.TypedNodeDisruption method*), 270
Scenario (*class in framework.scenario*), 277
scenario (*framework.scenario.ScenarioEnv property*), 279
scenario (*framework.tactics_helpers.dyn_generator_from_scenario attribute*), 266
scenario (*framework.tactics_helpers.DynGeneratorFromScenario attribute*), 264
scenario_ref_from() (*framework.tactics_helpers.Tactics static method*), 266
ScenarioEnv (*class in framework.scenario*), 279
sd_constraint_fuzz (*class in framework.generic_data_makers*), 224
sd_fuzz_separator_nodes (*class in framework.generic_data_makers*), 225
sd_fuzz_typed_nodes (*class in framework.generic_data_makers*), 225
sd_struct_constraints (*class in framework.generic_data_makers*), 227
sd_switch_to_alternate_conf (*class in framework.generic_data_makers*), 227
sd_walk_csp_solutions (*class in framework.generic_data_makers*), 228
sd_walk_data_model (*class in framework.generic_data_makers*), 228
SELECT() (*in module framework.dmhelpers.generic*), 287
semantics (*framework.node.Node attribute*), 167
send_data() (*framework.target_helpers.EmptyTarget method*), 229
send_data() (*framework.target_helpers.Target method*), 232
send_data() (*framework.targets.debug.TestTarget method*), 243
send_data() (*framework.targets.local.LocalTarget method*), 238
send_data() (*framework.targets.network.NetworkTarget method*), 237
send_data() (*framework.targets.printer.PrinterTarget method*), 241
send_data() (*framework.targets.ssh.SSHTarget method*), 240
send_data_and_log() (*framework.plumbing.FmkPlumbing method*), 300
send_data_sync() (*framework.target_helpers.Target method*), 232
send_multiple_data() (*framework.target_helpers.EmptyTarget method*), 230
send_multiple_data() (*framework.target_helpers.Target method*), 232
send_multiple_data() (*framework.targets.debug.TestTarget method*), 243
send_multiple_data() (*framework.targets.network.NetworkTarget method*), 237
send_multiple_data_sync() (*framework.target_helpers.Target method*), 232

send_pending_data() (*framework.target_helpers.Target method*), 232
sending_delay (*framework.scenario.Step property*), 281
sending_delay (*framework.target_helpers.Target attribute*), 232
Separator (*framework.dmhelpers.generic.MH.Attr attribute*), 284
Separator (*framework.node.NodeInternals attribute*), 176
SeparatorDisruption (*class in framework.fuzzing_primitives*), 269
Serial_Backend (*class in framework.comm_backends*), 259
set() (*framework.data.AttrGroup method*), 155
set_absorb_helper() (*framework.node.Node method*), 173
set_absorb_helper() (*framework.node.NodeInternals method*), 177
set_additional_info() (*framework.tactics_helpers.Tactics method*), 266
set_anchor() (*framework.scenario.Scenario method*), 278
set_attr() (*framework.node.Node method*), 173
set_attr() (*framework.node.NodeInternals method*), 177
set_attr() (*framework.tactics_helpers.Disruptor method*), 262
set_attr() (*framework.tactics_helpers.Generator method*), 264
set_attr() (*framework.tactics_helpers.StatefulDisruptor method*), 264
set_attributes_from() (*framework.data.Data method*), 157
set_attrs_from() (*framework.node.NodeInternals method*), 177
set_basic_attributes() (*framework.data.Data method*), 157
set_bitfield() (*framework.value_types.BitField method*), 201
set_bytes() (*framework.knowledge.feedback_collector.FeedbackCollector method*), 292
set_child_attr() (*framework.node.NodeInternals method*), 178
set_child_attr() (*framework.node.NodeInternals_GenFunc method*), 180
set_child_attr() (*framework.node.NodeInternals_NonTerm method*), 184
set_child_current_conf() (*framework.node.NodeInternals_GenFunc method*), 180
set_child_current_conf() (*framework.node.NodeInternals_NonTerm method*), 184
set_child_current_conf() (*framework.node.NodeInternals_Term method*), 185
set_child_env() (*framework.node.NodeInternals_Empty method*), 178
set_child_env() (*framework.node.NodeInternals_GenFunc method*), 180
set_child_env() (*framework.node.NodeInternals_NonTerm method*), 184
set_child_env() (*framework.node.NodeInternals_Term method*), 185
set_clone_info() (*framework.node.NodeInternals method*), 178
set_clone_info() (*framework.node.NodeInternals_Func method*), 179
set_clone_info() (*framework.node.NodeInternals_GenFunc method*), 180
set_clone_info() (*framework.node.NodeInternals_NonTerm method*), 184
set_comments() (*framework.operator_helpers.LastInstruction method*), 245
set_concrete_nodes() (*framework.node.NodeAbstraction method*), 175
set_consumer() (*framework.fuzzing_primitives.ModelWalker method*), 268
set_contents() (*framework.node.Node method*), 173
set_contents_from() (*framework.node.NodeInternals method*), 178
set_control_delay() (*framework.targets.debug.TestTarget method*), 243
set_control_over() (*framework.targets.debug.TestTarget method*), 243
set_csp() (*framework.node.Node method*), 174
set_current_conf() (*framework.node.Node method*), 174
set_data_id() (*framework.data.Data method*), 157
set_data_model() (*framework.data.Data method*), 157
set_data_model() (*framework.monitor.Monitor method*), 251
set_data_model() (*framework.node.Env method*), 165
set_data_model() (*framework.project.Project method*), 245
set_data_model() (*framework.scenario.Scenario*

method), 278
set_data_model() (*framework.target_helpers.Target method*), 232
set_default_value() (*framework.node.Node method*), 174
set_default_value() (*framework.value_types.BitField method*), 201
set_default_value() (*framework.value_types.INT method*), 207
set_default_value() (*framework.value_types.String method*), 215
set_default_value() (*framework.value_types.VT method*), 218
set_description() (*framework.value_types.String method*), 215
set_disruptor_weight() (*framework.plumbing.FmkPlumbing method*), 300
set_disruptor_weight() (*framework.tactics_helpers.Tactics method*), 266
set_dmaker_reset() (*framework.scenario.Step method*), 281
set_drawn_node_attrs() (*framework.node.Env4NT method*), 165
set_encoder() (*framework.node.NodeInternals_NonTerm method*), 184
set_env() (*framework.node.Node method*), 174
set_error() (*framework.plumbing.FmkPlumbing method*), 300
set_error_code() (*framework.knowledge.feedback_collector.FeedbackCollector method*), 292
set_exclusive_criteria() (*framework.node.NodeSemanticsCriteria method*), 187
set_exportable_fmk_ops() (*framework.project.Project method*), 245
set_exportable_fmk_ops() (*framework.tactics_helpers.DataMaker method*), 261
set_external_display() (*framework.logger.Logger method*), 249
Set_FbkTimeout (*framework.data.CallBackOps attribute*), 156
set_feedback_mode() (*framework.plumbing.FmkPlumbing method*), 300
set_feedback_mode() (*framework.target_helpers.Target method*), 232
set_feedback_timeout() (*framework.plumbing.FmkPlumbing method*), 300
set_feedback_timeout() (*framework.target_helpers.Target method*), 232
set_flag() (*framework.data.CallBackOps method*), 156
set_flag() (*framework.operator_helpers.Operation method*), 246
set_fmk_ops() (*framework.monitor.Monitor method*), 251
set_frozen_value() (*framework.node.Node method*), 174
set_func() (*framework.node.Node method*), 174
set_func_arg() (*framework.node.NodeInternals_Func method*), 179
set_fuzz_weight() (*framework.node.Node method*), 174
set_generator_func() (*framework.node.Node method*), 174
set_generator_func_arg() (*framework.node.NodeInternals_GenFunc method*), 180
set_generator_weight() (*framework.plumbing.FmkPlumbing method*), 300
set_generator_weight() (*framework.tactics_helpers.Tactics method*), 266
set_graph_info() (*framework.node.DynNode_Helpers method*), 164
set_health_check_timeout() (*framework.plumbing.FmkPlumbing method*), 300
set_history() (*framework.data.Data method*), 157
set_initial_dmaker() (*framework.data.Data method*), 157
set_instruction() (*framework.operator_helpers.LastInstruction method*), 245
set_internals() (*framework.node.Node method*), 174
set_items() (*framework.node.NodeCustomization method*), 176
set_logger() (*framework.monitor.Monitor method*), 251
set_logger() (*framework.project.Project method*), 245
set_logger() (*framework.target_helpers.Target method*), 232
set_mandatory_criteria() (*framework.node.NodeSemanticsCriteria method*), 187
set_monitor() (*framework.project.Project method*), 245
set_negative_criteria() (*framework.node.NodeSemanticsCriteria method*), 187
set_node_constraint() (*framework.node.NodeInternalsCriteria method*), 178
set_node_interest() (*framework.fuzzing_primitives.NodeConsumerStub method*), 269
set_node_sync() (*framework.node.NodeInternals method*), 178

set_operator_feedback() (*framework.operator_helpers.LastInstruction method*), 245
set_operator_status() (*framework.operator_helpers.LastInstruction method*), 246
set_optionalbut1_criteria() (*framework.node.NodeSemanticsCriteria method*), 187
set_post_args() (*framework.targets.local.LocalTarget method*), 238
set_pre_args() (*framework.targets.local.LocalTarget method*), 238
set_printer_name() (*framework.targets.printer.PrinterTarget method*), 241
set_private() (*framework.node.Node method*), 174
set_private() (*framework.node.NodeInternals method*), 178
set_private_info() (*framework.monitor.ProbeStatus method*), 256
set_probe_delay() (*framework.monitor.Monitor method*), 251
set_probe_delay() (*framework.monitor.ProbeUser method*), 257
set_probe_delay() (*framework.plumbing.FmkPlumbing method*), 300
set_project() (*framework.target_helpers.Target method*), 232
set_reinit_anchor() (*framework.scenario.Scenario method*), 278
set_scenario_env() (*framework.scenario.Scenario method*), 278
set_scenario_env() (*framework.scenario.Step method*), 281
set_scenario_env() (*framework.scenario.Transition method*), 282
set_seed() (*framework.generic_data_makers.sd_constraint_fuzz method*), 225
set_seed() (*framework.generic_data_makers.sd_fuzz_separator_nodes method*), 225
set_seed() (*framework.generic_data_makers.sd_fuzz_typed_nodes method*), 226
set_seed() (*framework.generic_data_makers.sd_struct_constraints method*), 227
set_seed() (*framework.generic_data_makers.sd_switch_to_altern_node method*), 227
set_seed() (*framework.generic_data_makers.sd_walk_csp_solutions method*), 228
set_seed() (*framework.generic_data_makers.sd_walk_data_model method*), 229
set_seed() (*framework.tactics_helpers.StatefulDisruptor method*), 264
set_semantics() (*framework.node.Node method*), 174
set_sending_burst_counter() (*framework.plumbing.FmkPlumbing method*), 300
set_sending_delay() (*framework.plumbing.FmkPlumbing method*), 300
set_sending_delay() (*framework.target_helpers.Target method*), 232
set_separator_node() (*framework.node.NodeInternals_NonTerm method*), 184
set_size_from_constraints() (*framework.node.Node method*), 174
set_size_from_constraints() (*framework.node.NodeInternals method*), 178
set_size_from_constraints() (*framework.node.NodeInternals_Func method*), 179
set_size_from_constraints() (*framework.node.NodeInternals_GenFunc method*), 180
set_size_from_constraints() (*framework.node.NodeInternals_NonTerm method*), 184
set_size_from_constraints() (*framework.node.NodeInternals_TypedValue method*), 186
set_size_from_constraints() (*framework.value_types.BitField method*), 201
set_size_from_constraints() (*framework.value_types.INT method*), 207
set_size_from_constraints() (*framework.value_types.String method*), 215
set_size_from_constraints() (*framework.value_types.VT method*), 218
set_size_on_source_node() (*framework.node.SyncSizeObj method*), 190
set_specific_fuzzy_values() (*framework.node.NodeInternals_TypedValue method*), 186
set_status() (*framework.operator_helpers.Operation method*), 246
set_subfield() (*framework.value_types.BitField method*), 201
set_subnode_default_qty() (*framework.node.NodeInternals_NonTerm method*), 184
set_subnode_minmax() (*framework.node.NodeInternals_NonTerm method*), 184
set_subnodes_basic() (*framework.node.Node method*), 174
set_subnodes_full_format() (*framework.node.Node method*), 174
set_subnodes_with_csts() (*framework.node.Node method*), 174

set_target() (*framework.scenario.Scenario method*), 279
set_target_ack_date() (*framework.logger.Logger method*), 249
set_target_ip() (*framework.targets.printer.PrinterTarget method*), 241
set_target_path() (*framework.targets.local.LocalTarget method*), 238
set_target_port() (*framework.targets.printer.PrinterTarget method*), 241
set_targets() (*framework.monitor.Monitor method*), 251
set_targets() (*framework.project.Project method*), 245
set_timeout() (*framework.comm_backends.SSH_Backend method*), 259
set_timeout() (*framework.targets.network.NetworkTarget method*), 237
set_timestamp() (*framework.monitor.ProbeStatus method*), 256
set_tmp_file_extension() (*framework.targets.local.LocalTarget method*), 238
set_tmp_file_extension() (*framework.targets.printer.PrinterTarget method*), 241
set_transitions() (*framework.scenario.Step method*), 281
set_values() (*framework.node.Node method*), 174
set_var_domain() (*framework.constraint_helpers.CSP method*), 297
setup() (*framework.generic_data_makers.d_add_data method*), 219
setup() (*framework.generic_data_makers.d_call_external_program method*), 219
setup() (*framework.generic_data_makers.d_corrupt_bits_by_range method*), 220
setup() (*framework.generic_data_makers.d_corrupt_node_bits method*), 220
setup() (*framework.generic_data_makers.d_fix_constraints method*), 221
setup() (*framework.generic_data_makers.d_fuzz_model_structure method*), 221
setup() (*framework.generic_data_makers.d_max_size method*), 221
setup() (*framework.generic_data_makers.d_modify_nodes method*), 222
setup() (*framework.generic_data_makers.d_next_node_content method*), 222
setup() (*framework.generic_data_makers.d_operate_on_nodes method*), 223
setup() (*framework.generic_data_makers.d_shallow_copy method*), 223
setup() (*framework.generic_data_makers.d_switch_to_alternate_conf method*), 224
setup() (*framework.generic_data_makers.g_generic_pattern method*), 224
setup() (*framework.generic_data_makers.g_population method*), 224
setup() (*framework.generic_data_makers.sd_constraint_fuzz method*), 225
setup() (*framework.generic_data_makers.sd_fuzz_separator_nodes method*), 225
setup() (*framework.generic_data_makers.sd_fuzz_typed_nodes method*), 226
setup() (*framework.generic_data_makers.sd_struct_constraints method*), 227
setup() (*framework.generic_data_makers.sd_switch_to_alternate_conf method*), 228
setup() (*framework.generic_data_makers.sd_walk_csp_solutions method*), 228
setup() (*framework.generic_data_makers.sd_walk_data_model method*), 229
setup() (*framework.tactics_helpers.Disruptor method*), 262
setup() (*framework.tactics_helpers.DynGenerator method*), 262
setup() (*framework.tactics_helpers.DynGeneratorFromScenario method*), 264
setup() (*framework.tactics_helpers.Generator method*), 264
setup() (*framework.tactics_helpers.StatefulDisruptor method*), 264
setup() (*libs.utils.Task method*), 302
SetupRequired (*framework.tactics_helpers.DataMakerAttr attribute*), 262
shallow_record() (*framework.logger.Logger method*), 249
share_knowlegde_source() (*framework.project.Project method*), 245
Shell_Backend (*class in framework.comm_backends*), 261
shmem_size (*framework.targets.debug.TestTarget attribute*), 243
ShmemMappingError, 242
show() (*framework.data.Data method*), 157
show() (*framework.data.DataBackend method*), 158
show() (*framework.data.NodeBackend method*), 160
show() (*framework.data.RawBackend method*), 160
show() (*framework.data_model.DataModel method*), 162
show() (*framework.node.Node method*), 175

show_and_flush_errors() (framework.plumbing.FmkPlumbing method), 300
show_atom_identifiers() (framework.plumbing.FmkPlumbing method), 300
show_data() (framework.plumbing.FmkPlumbing method), 300
show_data_bank() (framework.plumbing.FmkPlumbing method), 300
show_data_maker_types() (framework.plumbing.FmkPlumbing method), 300
show_data_models() (framework.plumbing.FmkPlumbing method), 300
show_disruptors() (framework.plumbing.FmkPlumbing method), 301
show_fmk_internals() (framework.plumbing.FmkPlumbing method), 301
show_generators() (framework.plumbing.FmkPlumbing method), 301
show_knowledge() (framework.plumbing.FmkPlumbing method), 301
show_operators() (framework.plumbing.FmkPlumbing method), 301
show_probes() (framework.plumbing.FmkPlumbing method), 301
show_projects() (framework.plumbing.FmkPlumbing method), 301
show_scenario() (framework.plumbing.FmkPlumbing method), 301
show_targets() (framework.plumbing.FmkPlumbing method), 301
show_tasks() (framework.plumbing.FmkPlumbing method), 301
show_wkspace() (framework.plumbing.FmkPlumbing method), 301
shrink_db() (framework.database.Database method), 275
SimilarityMeasure (class in framework.knowledge.feedback_handler), 293
SINT16_be (class in framework.value_types), 208
SINT16_le (class in framework.value_types), 209
SINT32_be (class in framework.value_types), 209
SINT32_le (class in framework.value_types), 209
SINT64_be (class in framework.value_types), 210
SINT64_le (class in framework.value_types), 210
SINT8 (class in framework.value_types), 210
size (framework.database.FeedbackGate property), 276
Size (framework.node.SyncScope attribute), 190
size (framework.value_types.INT attribute), 207
size (framework.value_types.INT16 attribute), 207
size (framework.value_types.INT32 attribute), 207
size (framework.value_types.INT64 attribute), 207
size (framework.value_types.INT8 attribute), 208
size() (framework.evolutionary_helpers.Population method), 291
size_for_absorption (framework.node.SyncSizeObj property), 190
sizesync_corrupt_hook() (framework.node.NodeInternals_NonTerm static method), 184
sources_names() (framework.database.FeedbackGate method), 276
split_verbose_with() (in module framework.node), 190
split_with() (in module framework.node), 190
SSH_Backend (class in framework.comm_backends), 258
SSHTarget (class in framework.targets.ssh), 239
standard (framework.dmhelpers.xml.TAG_TYPE attribute), 288
start() (framework.comm_backends.Backend method), 258
start() (framework.database.Database method), 275
start() (framework.knowledge.feedback_handler.FeedbackHandler method), 293
start() (framework.logger.Logger method), 249
start() (framework.monitor.Monitor method), 251
start() (framework.monitor.Probe method), 252
start() (framework.monitor.ProbeCmd method), 253
start() (framework.monitor.ProbeMem method), 254
start() (framework.monitor.ProbePID method), 256
start() (framework.monitor.ProbeUser method), 257
start() (framework.operator_helpers.Operator method), 246
start() (framework.plumbing.FmkPlumbing method), 301
start() (framework.plumbing.Printer method), 301
start() (framework.project.Project method), 245
start() (framework.target_helpers.Target method), 232
start() (framework.targets.debug.TestTarget method), 243
start() (framework.targets.local.LocalTarget method), 238
start() (framework.targets.network.NetworkTarget method), 237
start() (framework.targets.printer.PrinterTarget method), 241
start() (framework.targets.ssh.SSHTarget method), 241
start() (libs.utils.Term method), 302
start_new_log_entry() (framework.logger.Logger method), 249
start_new_shape() (framework.node_builder.RegexParser method), 199
start_new_shape_from_buffer() (framework.node_builder.RegexParser method), 199
start_probe() (framework.monitor.Monitor method), 251
Start_Task (framework.data.CallBackOps attribute),

156
start_term() (*libs.utils.ExternalDisplay method*), 302
State (*class in framework.node_builder*), 199
StatefulDisruptor (*class in framework.tactics_helpers*), 264
StateMachine (*class in framework.node_builder*), 199
status (*framework.monitor.Probe property*), 252
STATUS_THRESHOLD_FOR_RECOVERY (*framework.target_helpers.Target attribute*), 230
STATUS_THRESHOLD_FOR_RECOVERY (*framework.targets.ssh.SSHTarget attribute*), 239
Step (*class in framework.scenario*), 279
step (*framework.scenario.Transition property*), 282
steps (*framework.scenario.Scenario property*), 279
StepStub (*class in framework.scenario*), 281
stick_to_default_mode (*framework.node.NonTermCusto property*), 188
StickToDefault (*framework.dmhelpers.generic.MH.Custo.NTerm attribute*), 285
StickToDefault (*framework.node.NonTermCusto attribute*), 188
still_interested_by() (*framework.fuzzing_primitives.AltConfConsumer method*), 267
still_interested_by() (*framework.fuzzing_primitives.NodeConsumerStub method*), 269
still_interested_by() (*framework.fuzzing_primitives.NonTermVisitor method*), 269
still_interested_by() (*framework.fuzzing_primitives.TypedNodeDisruption method*), 270
Stop (*framework.operator_helpers.Operation attribute*), 246
stop() (*framework.comm_backends.Backend method*), 258
stop() (*framework.database.Database method*), 275
stop() (*framework.knowledge.feedback_handler.FeedbackHandler method*), 293
stop() (*framework.logger.Logger method*), 249
stop() (*framework.monitor.BlockingProbeUser method*), 250
stop() (*framework.monitor.Monitor method*), 251
stop() (*framework.monitor.Probe method*), 252
stop() (*framework.monitor.ProbeCmd method*), 253
stop() (*framework.monitor.ProbeMem method*), 254
stop() (*framework.monitor.ProbePID method*), 256
stop() (*framework.monitor.ProbeUser method*), 257
stop() (*framework.operator_helpers.Operator method*), 246
stop() (*framework.plumbing.FmkPlumbing method*), 301
stop() (*framework.plumbing.FmkTask method*), 301
stop() (*framework.plumbing.Printer method*), 302
stop() (*framework.project.Project method*), 245
stop() (*framework.target_helpers.Target method*), 233
stop() (*framework.targets.debug.TestTarget method*), 243
stop() (*framework.targets.local.LocalTarget method*), 238
stop() (*framework.targets.network.NetworkTarget method*), 237
stop() (*framework.targets.ssh.SSHTarget method*), 241
stop() (*libs.utils.ExternalDisplay method*), 302
stop() (*libs.utils.Term method*), 302
stop_all_probes() (*framework.monitor.Monitor method*), 251
stop_all_probes() (*framework.plumbing.FmkPlumbing method*), 301
stop_all_tasks() (*framework.plumbing.FmkPlumbing method*), 301
stop_probe() (*framework.monitor.Monitor method*), 251
stop_probe() (*framework.plumbing.FmkPlumbing method*), 301
Stop_Task (*framework.data.CallBackOps attribute*), 156
StopProcessingCB (*framework.data.CallBackOps attribute*), 156
String (*class in framework.value_types*), 211
structure_will_change() (*framework.node.NodeInternals_NonTerm method*), 184
subclass_fuzzing_list (*framework.value_types.String attribute*), 211
subclass_specific_init() (*framework.value_types.Filename method*), 204
subclass_specific_init() (*framework.value_types.String method*), 215
subclass_specific_test_cases() (*framework.value_types.Filename method*), 204
subclass_specific_test_cases() (*framework.value_types.FolderPath method*), 204
subclass_specific_test_cases() (*framework.value_types.String method*), 215
submit_sql_stmt() (*framework.database.Database method*), 275
supported_feedback_mode (*framework.target_helpers.EmptyTarget attribute*), 230
supported_feedback_mode (*framework.target_helpers.Target attribute*), 233
supported_feedback_mode (*framework.targets.debug.TestTarget attribute*), 243
supported_feedback_mode (*framework.targets.local.LocalTarget attribute*),

238
supported_feedback_mode (*framework.targets.network.NetworkTarget attribute*), 237
supported_feedback_mode (*framework.targets.printer.PrinterTarget attribute*), 241
switch_feedback_mode() (*framework.plumbing.FmkPlumbing method*), 301
switch_mode() (*framework.value_types.VT_Alt method*), 218
switch_term() (*framework.plumbing.FmkPlumbing method*), 301
SyncExistenceObj (*class in framework.node*), 188
synchronize_nodes() (*framework.node.NodeInternals method*), 178
synchronize_nodes() (*framework.node.SyncObj method*), 189
synchronized_with() (*framework.node.Node method*), 175
SyncObj (*class in framework.node*), 189
SyncQtyFromObj (*class in framework.node*), 189
SyncScope (*class in framework.node*), 189
SyncSizeObj (*class in framework.node*), 190

T

Tactics (*class in framework.tactics_helpers*), 265
tag_builder() (*in module framework.dmhelpers.xml*), 288
TAG_TYPE (*class in framework.dmhelpers.xml*), 288
take_info_ownership() (*framework.data.Data method*), 157
Target (*class in framework.target_helpers*), 230
target (*framework.scenario.ScenarioEnv property*), 279
target_status (*framework.monitor.Monitor property*), 251
TargetError, 233
TargetNotReady, 233
targets (*libs.utils.Task attribute*), 302
TargetStuck, 233
Task (*class in libs.utils*), 302
tasks_to_start (*framework.scenario.Step property*), 281
tasks_to_stop (*framework.scenario.Scenario property*), 279
tasks_to_stop (*framework.scenario.Step property*), 281
Term (*class in libs.utils*), 302
terminate() (*framework.targets.local.LocalTarget method*), 238
terminate() (*framework.targets.network.NetworkTarget method*), 237
Test (*class in framework.knowledge.information*), 296
TestFbkHandler (*class in framework.knowledge.feedback_handler*), 294
TestTarget (*class in framework.targets.debug*), 242
tg_ids (*framework.data.Data property*), 157
threshold (*framework.monitor.ProbeMem attribute*), 253, 254
timeout (*framework.monitor.ProbeTimeoutError property*), 256
timeout (*framework.monitor.ProbeUser attribute*), 257
TIMESTAMP() (*in module framework.dmhelpers.generic*), 287
tmp_ref_count (*framework.node.Node attribute*), 167
to_ascii() (*framework.node.Node method*), 175
to_bytes() (*framework.data.Data method*), 157
to_bytes() (*framework.data.DataBackend method*), 158
to_bytes() (*framework.data.EmptyBackend method*), 159
to_bytes() (*framework.data.NodeBackend method*), 160
to_bytes() (*framework.data.RawBackend method*), 160
to_bytes() (*framework.node.Node method*), 175
to_formatted_str() (*framework.data.Data method*), 157
to_formatted_str() (*framework.data.EmptyBackend method*), 159
to_formatted_str() (*framework.data.NodeBackend method*), 160
to_formatted_str() (*framework.data.RawBackend method*), 160
to_formatted_str() (*framework.node.Node method*), 175
to_str() (*framework.data.Data method*), 157
to_str() (*framework.data.DataBackend method*), 158
to_str() (*framework.data.EmptyBackend method*), 159
to_str() (*framework.data.NodeBackend method*), 160
to_str() (*framework.data.RawBackend method*), 160
to_str() (*framework.node.Node method*), 175
tolerance (*framework.monitor.ProbeMem attribute*), 253, 254
transform_func (*framework.node.NodeCustomization property*), 176
Transition (*class in framework.scenario*), 282
transitions (*framework.scenario.Scenario property*), 279
transitions (*framework.scenario.Step property*), 281
trigger_feedback_handlers() (*framework.project.Project method*), 245
trigger_last_mode (*framework.node.GenFuncCusto property*), 166
TriggerLast (*framework.dmhelpers.generic.MH.Custo.Gen attribute*), 285
TriggerLast (*framework.node.GenFuncCusto attribute*), 166

truncate_info() (*in module framework.generic_data_makers*), 229
trust_level (*framework.knowledge.information.Info property*), 294
trust_value (*framework.knowledge.information.Info property*), 294
TrustLevel (*class in framework.knowledge.information*), 296
TypedNodeDisruption (*class in framework.fuzzing_primitives*), 270

U

UINT16_be (*class in framework.value_types*), 215
UINT16_le (*class in framework.value_types*), 215
UINT32_be (*class in framework.value_types*), 216
UINT32_le (*class in framework.value_types*), 216
UINT64_be (*class in framework.value_types*), 216
UINT64_le (*class in framework.value_types*), 216
UINT8 (*class in framework.value_types*), 217
unfreeze() (*framework.node.Node method*), 175
unfreeze() (*framework.node.NodeInternals_GenFunc method*), 180
unfreeze() (*framework.node.NodeInternals_NonTerm method*), 184
unfreeze() (*framework.node.NodeInternals_Term method*), 185
unfreeze_all() (*framework.node.Node method*), 175
unfreeze_all() (*framework.node.NodeInternals_GenFunc method*), 180
unfreeze_all() (*framework.node.NodeInternals_NonTerm method*), 184
unfreeze_all() (*framework.node.NodeInternals_Term method*), 185
UNICODE (*framework.dmhelpers.generic.MH.Charset attribute*), 285
Unknown (*framework.knowledge.information.Hardware attribute*), 294
Unknown (*framework.knowledge.information.InputHandling attribute*), 295
Unknown (*framework.knowledge.information.Language attribute*), 295
Unknown (*framework.knowledge.information.OS attribute*), 295
UNKNOWN_SEMANTIC (*framework.targets.network.NetworkTarget attribute*), 233
unplan_reset() (*framework.node.NodeInternals_NonTerm.NodeAttrs method*), 181
unroll() (*framework.node.NodeInternals_NonTerm.NodeAttrs method*), 181
update() (*framework.node.Node method*), 175
update_atom() (*framework.data_model.DataModel method*), 162
update_atom() (*framework.data_model.NodeBackend method*), 163
update_from() (*framework.data.Data method*), 157
update_from() (*framework.data.DataBackend method*), 158
update_from() (*framework.data.NodeBackend method*), 160
update_from() (*framework.data.RawBackend method*), 160
update_node_ids() (*framework.node.Env4NT method*), 165
update_node_refs() (*framework.node.Env method*), 165
update_raw_value() (*framework.value_types.BitField method*), 202
update_raw_value() (*framework.value_types.INT method*), 207
update_value() (*framework.node.NodeInternals_Term method*), 185
uri_prefix (*framework.value_types.Filename attribute*), 204
uri_suffix (*framework.value_types.Filename attribute*), 204
usable (*framework.value_types.INT attribute*), 207
usable (*framework.value_types.INT16 attribute*), 207
usable (*framework.value_types.INT32 attribute*), 207
usable (*framework.value_types.INT64 attribute*), 207
usable (*framework.value_types.INT8 attribute*), 208
usable (*framework.value_types.INT_str attribute*), 208
usable (*framework.value_types.SINT16_be attribute*), 209
usable (*framework.value_types.SINT16_le attribute*), 209
usable (*framework.value_types.SINT32_be attribute*), 209
usable (*framework.value_types.SINT32_le attribute*), 210
usable (*framework.value_types.SINT64_be attribute*), 210
usable (*framework.value_types.SINT64_le attribute*), 210
usable (*framework.value_types.SINT8 attribute*), 211
usable (*framework.value_types.UINT16_be attribute*), 215
usable (*framework.value_types.UINT16_le attribute*), 216
usable (*framework.value_types.UINT32_be attribute*), 216
usable (*framework.value_types.UINT32_le attribute*), 216
usable (*framework.value_types.UINT64_be attribute*), 216

usable (*framework.value_types.UINT64_le attribute*), 217
usable (*framework.value_types.UINT8 attribute*), 217
user_context (*framework.scenario.Scenario property*), 279
user_context (*framework.scenario.ScenarioEnv property*), 279
UTF16BE (*framework.value_types.String attribute*), 211
UTF16LE (*framework.value_types.String attribute*), 211

V

valid_keys (*framework.node_builder.NodeBuilder attribute*), 192
validation_tests() (*framework.data_model.DataModel method*), 162
value (*framework.knowledge.feedback_handler.SimilarityMeasure property*), 294
value (*framework.monitor.ProbeStatus property*), 256
value_space_size (*framework.value_types.INT attribute*), 207
value_space_size (*framework.value_types.INT16 attribute*), 207
value_space_size (*framework.value_types.INT32 attribute*), 207
value_space_size (*framework.value_types.INT64 attribute*), 208
value_space_size (*framework.value_types.INT8 attribute*), 208
value_space_size (*framework.value_types.INT_str attribute*), 208
var_domain (*framework.constraint_helpers.Constraint property*), 298
var_domain_updated (*framework.constraint_helpers.CSP property*), 297
var_mapping (*framework.constraint_helpers.CSP property*), 297
vars (*framework.constraint_helpers.Constraint attribute*), 298
VERYLOW_PRIO (*framework.node_builder.NodeBuilder attribute*), 190
VT (*class in framework.value_types*), 217
VT_Alt (*class in framework.value_types*), 218

W

wait_for_exhaustion() (*framework.fuzzing_primitives.AltConfConsumer method*), 267
wait_for_exhaustion() (*framework.fuzzing_primitives.BasicVisitor method*), 267
wait_for_exhaustion() (*framework.fuzzing_primitives.NodeConsumerStub method*), 269
wait_for_exhaustion() (*framework.fuzzing_primitives.NonTermVisitor method*), 269
wait_for_probe_init() (*framework.monitor.ProbeUser method*), 257
wait_for_probe_initialization() (*framework.monitor.Monitor method*), 251
wait_for_probe_status_retrieval() (*framework.monitor.Monitor method*), 251
wait_for_sync() (*framework.logger.Logger method*), 249
wait_for_sync() (*framework.plumbing.Printer method*), 302
wait_for_target_readiness() (*framework.plumbing.FmkPlumbing method*), 301
wait_until_armed() (*framework.monitor.BlockingProbeUser method*), 250
wait_until_ready() (*framework.monitor.BlockingProbeUser method*), 250
walk() (*framework.node.Node method*), 175
walk_graph_rec() (*framework.fuzzing_primitives.ModelWalker method*), 268
walk_to() (*framework.scenario.Scenario method*), 279
walk_to_reinit() (*framework.scenario.Scenario method*), 279
what_match_from() (*framework.node.NodeSemantics method*), 187
Windows (*framework.knowledge.information.OS attribute*), 295
windows_prefix (*framework.value_types.Filename attribute*), 204
windows_specific_fnames (*framework.value_types.Filename attribute*), 204
windows_suffix (*framework.value_types.Filename attribute*), 204
wkspace_enabled (*framework.project.Project attribute*), 245
wkspace_free_slot_ratio_when_full (*framework.project.Project attribute*), 245
wkspace_size (*framework.project.Project attribute*), 245
WRAP() (*in module framework.dmhelpers.generic*), 287
Wrap_Enc (*class in framework.encoders*), 273
Wrapper (*class in framework.value_types*), 218
write() (*framework.logger.Logger method*), 249
write() (*framework.plumbing.Printer method*), 302
WRITE_API (*framework.logger.Logger attribute*), 247

X

X86_32 (*framework.knowledge.information.Hardware attribute*), 294

X86_64 (*framework.knowledge.information.Hardware attribute*), 294
xml_decl_builder() (*in module framework.dmhelpers.xml*), 289

Z

ZeroCopy (*framework.dmhelpers.generic.MH attribute*), 286