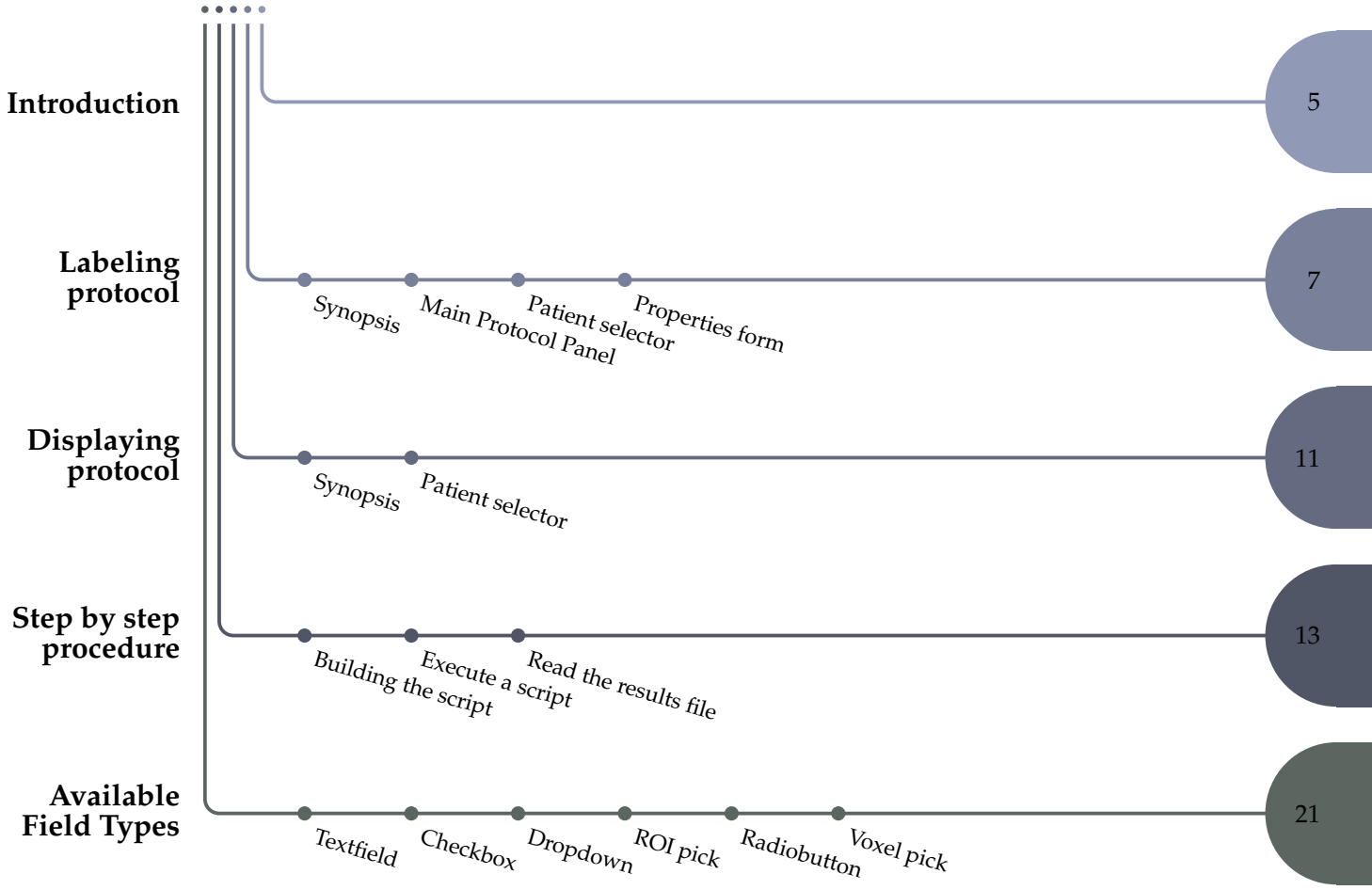
User guide

Labeling application
Displaying application
— LIFEx —

C. Nioche, F. Orlhac, I. Buvat

LIFEx version 26.5.n,
Last update of document: 2026/05/20


Introduction
5
Labeling protocol
7
Synopsis
Main Protocol Panel
Patient selector
Properties form
Displaying protocol
11
Synopsis
Patient selector
Step by step procedure
13
Building the script
Execute a script
Read the results file
Available Field Types
21
Textfield
Checkbox
Dropdown
ROI pick
Radiobutton
Voxel pick

Chapter 1
Introduction

Labeling protocol - introduction The labeling protocol allows users to annotate images and related regions with ease.

It is a generic module that lets you define predefined question/answer sets for application-specific annotation and populate a database. This database can then be used for machine learning or deep learning.

The questions (and possible answers) are defined in a user-created script. This script specifies all the information required for the annotation task. Running the script automates the workflow: images are loaded and the relevant annotation menus are presented automatically.

Progress is tracked by a controller that monitors and displays how far the script has advanced, so users can check their progress at any time.

The protocol uses three panels opened in LIFEx at three locations:

1. the controller in the protocol master panel (blue window at the top, next to other protocols),
2. the patient selector (blue window on the left, in the patient panel),
3. the properties input panel (blue window on the right, next to the ROI panel).

These three panels and their actions are described in the following sections.

Displaying protocol - introduction The display protocol is a simplified variant of the labeling protocol. It chains series/ROI reading without recording properties.

Chapter 2
Labeling protocol

2.1 Synopsis

The labeling protocol consists of three input panels (blue across the application). They are displayed after loading a script file. Figure 2.1 (p. 8) shows an example UI when the protocol is running.

2.2 Main Protocol Panel

Main Protocol Window. This panel explains the steps to follow during annotation.

Reset button. A reset button is provided. It deletes the annotation results file. This does *not* affect saved ROI files: those files remain on disk.

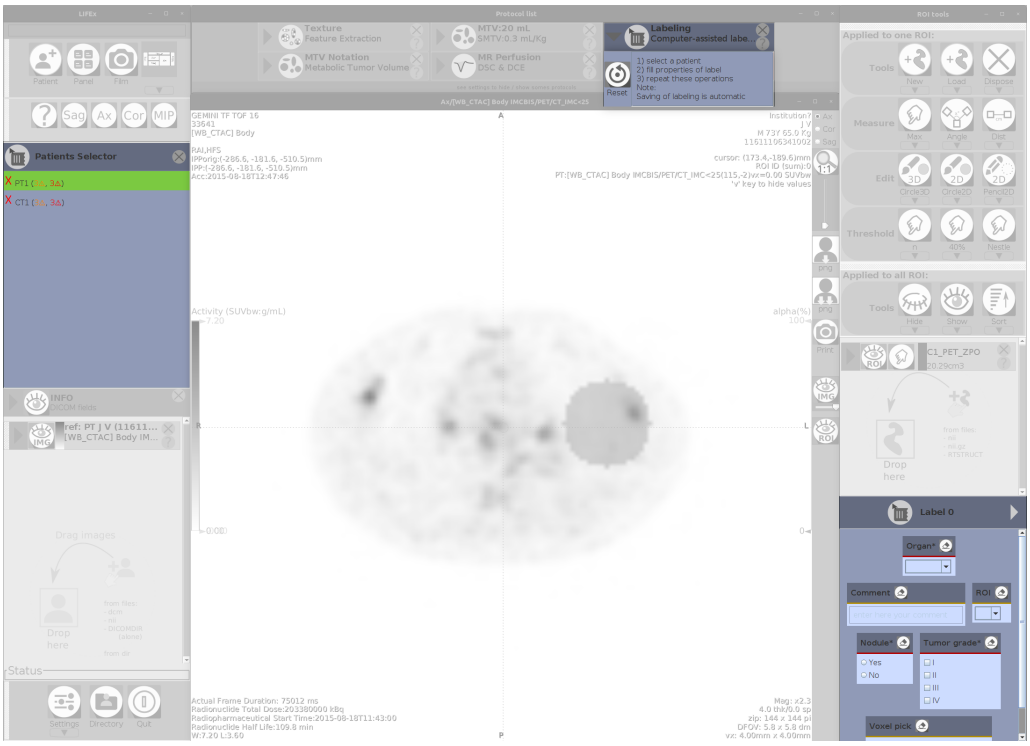


Figure 2.1: Main in GUI

2.3 Patient selector

The list of patients defined in the script are indexed and displayed in the patient selector list. These are the file names that will be retrieved and displayed for the annotation process.

The list also shows progress using red crosses and green check marks. Each patient with incomplete annotation appears with a red cross (Figure 2.3, p. 9). Patients fully annotated appear with a green check mark (Figure 2.4, p. 10).

To the right of the patient's name in this list are the missing data and warnings associated with the completeness of the annotation. The "warnings" are displayed in orange color and correspond to the number of fields not filled in but that are not mandatory. The "missing data" are displayed in red color and correspond to the mandatory fields that are not filled in (figure 2.3, p.9).

2.4 Properties form

2.4.1 Graphic user interface of properties

Form. The properties form is built from the definitions in the script. Each definition creates a property and its input widget.

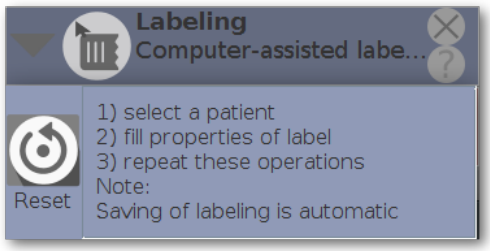


Figure 2.2: ProtocolGui

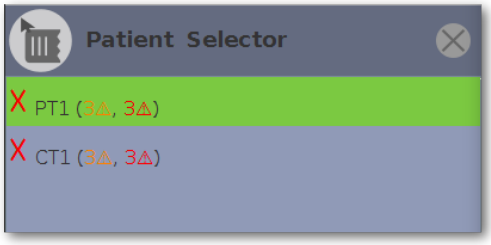


Figure 2.3: Patient Selector with showing the patients that still need to be annotated

Label. Properties are grouped under a label. You can add as many labels as needed. Navigation buttons (left/right arrows) beside the label title let you browse Label 0, Label 1, and so on.

Note. All properties under a label must be completed to mark the patient as fully annotated (green check mark).

2.4.2 Three audit levels = three color levels

The form uses three visual levels to verify fields:

1. **Mandatory field missing** (red line): the field is mandatory (title marked with *) and is empty;
2. **Warning** (orange line): non-mandatory field is empty;
3. **OK** (green line): the field is properly completed.

Figure 2.4: Patient Selector showing that all patients have been annotated

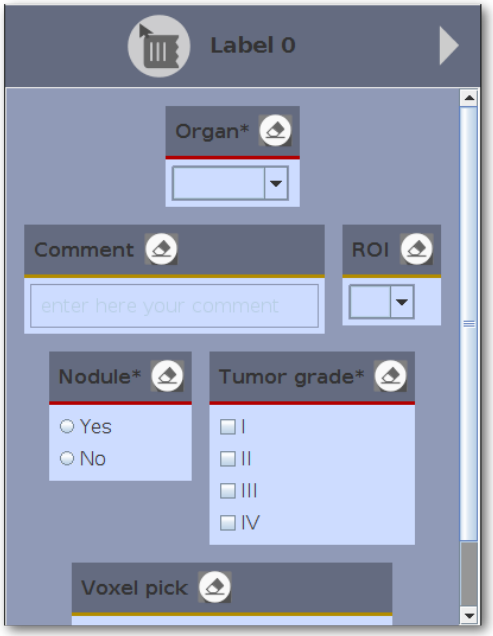


Figure 2.5: Properties GUI

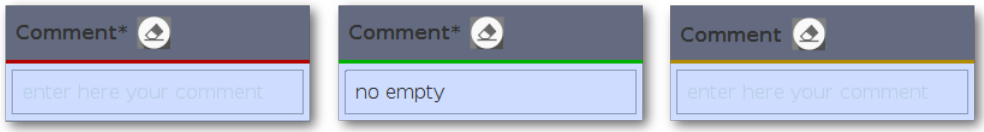


Figure 2.6: Examples: missing (red), OK (green), needs attention (orange)

Chapter 3
Displaying protocol

3.1 Synopsis

The displaying protocol consists of a single input panel (blue), shown after loading a script file. This corresponding script allows you to easily display a pre-defined patient list with their ROI)

3.2 Patient selector

The list of patients defined in the script is indexed and displayed in the selector. These are the file names that will be loaded (Figure 3.1, p. 12).

Figure 3.1: Patient Selector showing that all patients have been annotated

Chapter 4
Step by step procedure

4.1 Building the script

What is a script? A script file is a human-readable text file. Each line uses the syntax `key=value`.

In the line `LIFEx.script=labeling`, the key is `script` and the value is `labeling`. All properties in the script follow this syntax.

Understanding the different sections of the script. The script has several sections. The first `Common` section is fixed and must not be changed. It tells the application what type of script follows.

4.1.1 Fixed and common section

`LIFEx.Script=labeling`
→ Script is for labeling

`LIFEx.Script=displaying`
→ Script is for displaying

Choose one of these two options: *labeling* or *displaying*.

Copy the common, fixed section as is. Do not modify it. It allows the application to interpret the rest of the script.

4.1.2 Properties section

`#` denotes the property index. It starts at 0 and is incremented by 1 for each new property.

`LIFEx.property#.title`
→ Property label (add ∗ when an answer is mandatory).
Example (mandatory tumor location for `property0`):
`LIFEx.property0.title=Tumor location*`

`LIFEx.property#.title.tooltiptext`
→ Optional tooltip text for the property label.

`LIFEx.property#.type`
→ Mandatory. One of `radiobutton | checkbox | textfield | dropdown | voxelPick | roiPick`.
Example (checkbox for `property0`): `LIFEx.property0.type=checkbox`.
See Chapter 5, p. 21, for details on field types.

`LIFEx.property#.value`
→ Mandatory list of selectable values, separated by `|`.
Example (tumor grade I-IV for `property0`): `LIFEx.property0.value=I|II|III|IV`.

`LIFEx.property#.value.tooltiptext`
→ Optional tooltip(s) for the values, also separated by `|`.
Example:
`LIFEx.property0.value.tooltiptext=Select I grade|Select II grade|Select III grade|Select IV grade`

4.1.3 Patient/Image section

From nifti files:
`LIFEx.patient#.series#=.../file`
→ mandatory, is the path to the image file to be read. It may be a nifti file. For instance, `LIFEx.patient0.series0=/home/user/I1.nii` to select the first patient (patient0) and read series I1 (series0) for that patient from the /home/user directory.

From directory with dicom files:
`LIFEx.patient#.series#=.../directory`
→ mandatory, is the path to the image file to be read. It may be a directory that contains files. For instance: `LIFEx.patient0.series0=/home/user/P1` to select the first patient (patient0) and read the first series (series0) in the directory P1 of /home/user.

From DICOMDIR file:
`LIFEx.patient#.series#=.../DICOMDIR`
→ mandatory, is the DICOM file.

4.1.4 Optional section

ROI:set Localisation On Views
`# add get button on all voxelpick properties -> optional [ true | false ], default: true LIFEx.properties.voxelpick.getLocalisationOnViews = false`
→ this true/false property allows you to add (or not) an additional button to locate the coordinates entered in this field on the slice plan. When this button is pressed, the display cross is placed on these coordinates and the views are updated.

ROI:directory of ROI of all patients-> optional
`LIFEx.roi=/new path/...`
→ is the directory where ROI files will be saved. This is optional because if it is not defined, the directory set in the application will be reused (see `Settings\ Common\ Data directory`). The path and filename must respect the syntax of the system on which you are running the application. With Windows system, the syntax C:/directory/home should be used for which the \ are translated into /.

ROI:directory of ROI of one patient -> optional
`LIFEx.patient#.roi=/new path/...`
is the directory where ROI files of patient # will be saved. This is optional because if it is not defined, the directory set in the application will be reused (see `Settings\ Common\ Data directory`). The path and filename must respect the syntax of the system on which you are running the application. With Windows system, the syntax C:/directory/home should be used for which the \ are translated into /.

ROI:saving -> optional You can prohibit/allow automatic saving:
`LIFEx.roi.saving=true || false`

Series: Window Leveling -> optional
```
# Window Leveling -> optional
LIFEx.all.patient.series0.window.width=10
LIFEx.all.patient.series0.window.level=5
```
→ It is possible to voluntarily set the windows/Leveling at certain values. To do this, simply fill in these values in the corresponding fields

```
# color map ; available options
# MONOCHROME1 || MONOCHROME1_PVALUES || MONOCHROME1_PVALUES0
# MONOCHROME2 || MONOCHROME2_PVALUES || MONOCHROME2_PVALUES0
```

```
# RAINBOW || RAINBOW_PVALUES || RAINBOW_PVALUES0
# RAINBOW_INVERSE || RAINBOW_INVERSE_PVALUES || RAINBOW_INVERSE_PVALUES0
# RAINBOW_NEW || RAINBOW_NEW_PVALUES || RAINBOW_NEW_PVALUES0
# RAINBOW_NEW_INVERSE || RAINBOW_NEW_INVERSE_PVALUES
# || RAINBOW_NEW_INVERSE_PVALUES0
# HEAT || HEAT_PVALUES || HEAT_PVALUES0
# HEAT_INVERSE || HEAT_INVERSE_PVALUES || HEAT_INVERSE_PVALUES0

# color map in first 3 frames -> optional, default:  MONOCHROME2
LIFEx.frame0.series.color=MONOCHROME2
LIFEx.frame1.series.color=MONOCHROME2
LIFEx.frame2.series.color=MONOCHROME2
```

→ You can set color palettes per frame (frame0, frame1, frame2). Use one of the names listed above.

Step by step procedure

4.1.5 Result section

`LIFEx.Output.File=.../LabelingResults`
→ Output file base name for annotation results. Paths must follow your OS syntax. On Windows, use `C:/directory/home` (backslashes are translated to `/`).

Two files are created: one `.props` (properties format) and one `.csv` (comma-separated values).

1. .../LabelingResults.props
2. .../LabelingResults.csv

Both files contain exactly the same data; only the format differs. Use the one you prefer.

4.1.6 Full example

```
# Common
# fixed information on script -> mandatory
LIFEx.script=labeling

# result file -> mandatory
LIFEx.Output.File=.../LabelingResults

# directory of ROI files -> optional
LIFEx.Output.Roi.File=.../roi

# is heading saved (in addition to the values) -> optional
# [ true | false ], default:  false
LIFEx.Output.Title.Saved=false

# Window Leveling -> optional
#LIFEx.all.patient.series0.window.width=10
```

```
#LIFEx.all.patient.series0.window.level=5
# color map ; available options
# MONOCHROME1 || MONOCHROME1_PVALUES || MONOCHROME1_PVALUES0
# MONOCHROME2 || MONOCHROME2_PVALUES || MONOCHROME2_PVALUES0
# RAINBOW || RAINBOW_PVALUES || RAINBOW_PVALUES0
# RAINBOW_INVERSE || RAINBOW_INVERSE_PVALUES || RAINBOW_INVERSE_PVALUES0
# RAINBOW_NEW || RAINBOW_NEW_PVALUES || RAINBOW_NEW_PVALUES0
# RAINBOW_NEW_INVERSE || RAINBOW_NEW_INVERSE_PVALUES
# || RAINBOW_NEW_INVERSE_PVALUES0
# HEAT || HEAT_PVALUES || HEAT_PVALUES0
# HEAT_INVERSE || HEAT_INVERSE_PVALUES || HEAT_INVERSE_PVALUES0

# color map in first 3 frames -> optional, default:  MONOCHROME2
LIFEx.frame0.series.color=MONOCHROME2
LIFEx.frame1.series.color=MONOCHROME2
LIFEx.frame2.series.color=MONOCHROME2


# properties
#
# Number of properties
# must be incremented by 1, start from 0

# LIFEx.property#.title -> is heading of property (add * when an answer is
absolutely required)
# LIFEx.property#.title.tooltiptext -> optional, is tool tip text added to
heading property
# LIFEx.property#.type -> mandatory [ radiobutton | checkbox | textfield
| dropdown | voxelpick | roipick ]
# LIFEx.property#.value -> mandatory, is property values available to be
selected from, several values are available with "|" character separator
# LIFEx.property#.value.tooltiptext -> optional, is tool tip text added to
value property, several texts are available with "|" character separator

# see examples, below:
# textfield
LIFEx.property0.title = Organ*
LIFEx.property0.type = dropdown
LIFEx.property0.value = Lung|Kidney|Stomach|Liver

# comment
LIFEx.property1.title = Comment*
LIFEx.property1.type = textfield

# roipick
LIFEx.property2.title = ROI
LIFEx.property2.type = roipick

# radiobutton
LIFEx.property3.title = Nodule*
LIFEx.property3.title.tooltiptext = This is the tool tip text corresponding
to the nodule heading
```

```
LIFEx.property3.type = radiobutton
LIFEx.property3.value = Yes|No
LIFEx.property3.value.tooltiptext = presence of a nodule|no nodule

# checkbox
LIFEx.property4.title = Tumor grade*
LIFEx.property4.type = checkbox
LIFEx.property4.value = I|II|III|IV

# voxelpick
LIFEx.property5.title = Voxel pick
LIFEx.property5.type = voxelpick

#
# image of patients or directory
LIFEx.patient0.series0=.../I0
LIFEx.patient1.series0=.../I1
LIFEx.patient2.series0=.../I2
```


4.2 Execute a script

Drag and drop the script file onto the patient loading area on the left. The application then starts automatically.

4.3 Read the results file

The results are stored in the file specified by `LIFEx.output.file`. Its syntax matches the script's: one `key=value` pair per line.

Example of results file:

```
#LIFExv.v.v Labeling properties
#Thu Apr 16 19:40:08 CEST 2020
LIFEx.patient0.label0.property0.value=Stomach
LIFEx.patient0.label0.property1.value=comment
LIFEx.patient0.label0.property2.value=
LIFEx.patient0.label0.property3.value=Yes
LIFEx.patient0.label0.property4.value=III
LIFEx.patient0.label0.property5.value=t0 z0 y61 x62
LIFEx.patient1.label0.property0.value=Lung
LIFEx.patient1.label0.property1.value=another comment
LIFEx.patient1.label0.property2.value=
LIFEx.patient1.label0.property3.value=No
LIFEx.patient1.label0.property4.value=IV
LIFEx.patient1.label0.property5.value=t0 z0 y66 x23
```

Explanation of the example file. This file contains annotations for two patients (`patient0` and `patient1`). Each has one label (`label0`). Values for six properties (`property0`-`property5`) are recorded without their labels.

You can open this file in Excel by using = (optionally with `.`) as the delimiter.

Chapter 5
Available Field Types

Several field types are available to build a complete form: *textfield*, *checkbox*, *dropdown*, *radiobutton*, *voxel pick*, and *ROI dropdown*. They are described below.

5.1 Textfield

The `textfield` (`LIFEx.property#.type=textfield`) is a standard form control with a label, input, and help text. It lets users enter and edit text (Figure 5.1, p. 22).

5.2 Checkbox

A `checkbox` (`LIFEx.property#.type=checkbox`) appears as a square box that is ticked when selected. Use it to let users choose one or more options from a small set (Figure 5.2, p. 22).

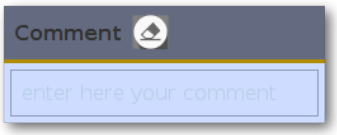


Figure 5.1: textfield type

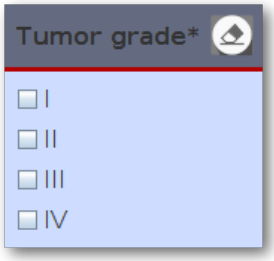


Figure 5.2: checkbox type

5.3 Dropdown

A `dropdown` (`LIFEx.property#.type=dropdown`) is a list of choices that opens under a menu title and remains until used or dismissed (Figure 5.3, p. 22).

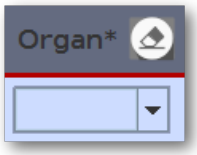


Figure 5.3: dropdown type

5.4 ROI pick

`roipick` (`LIFEx.property#.type=roipick`) displays a dropdown list of ROIs (Figure 5.4, p. 23).

5.5 Radiobutton

`radiobutton` (`LIFEx.property#.type=radiobutton`) controls are presented in groups of mutually exclusive options-only one can be selected at a time (Figure 5.5, p. 23).

Figure 5.4: roipick type

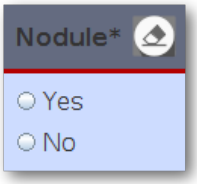


Figure 5.5: radiobutton type

5.6 Voxel pick

`voxelpick` (`LIFEx.property#.type=voxelpick`) lets users enter voxel coordinates *in millimetres* by mouse click. The field provides two buttons: *Get* (capture current coordinates) and *Set* (revisit the stored location when the patient is reopened) (Figure 5.6, p. 23).

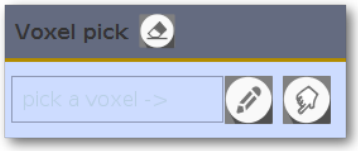


Figure 5.6: voxelpick type