tryton Documentation

Release 5.5

Anne Krings, Bertrand Chenal, Cédric Krier, Mathias Behrle, Tobia

Mar 27, 2020

Contents

Tryton is a Graphical User Interface to Tryton based on GTK and Python.

CHAPTER 1

Contents

1.1 Installing tryton

1.1.1 Installation

Once you've downloaded and unpacked a tryton source release, enter the directory where the archive was unpacked, and run:

```
python setup.py install
```

Note that you may need administrator/root privileges for this step, as this command will by default attempt to install tryton to the Python site-packages directory on your system.

For advanced options, please refer to the easy_install and/or the distutils documentation:

To use without installation, run `bin/tryton` from where the archive was unpacked.

1.2 Client Usage

This document is the reference about the concepts of the graphical user interface (also known as *Tryton client*) of the Tryton application framework.

1.2.1 Name

tryton - Graphical user client of the Tryton application framework

1.2.2 Synopsis

```
tryton [options] [url]
```

On startup the login dialog is displayed.

1.2.3 Options

--version Show program version number and exit

-h, --help Show help message and exit

-c FILE, --config=FILE Specify alternate *configuration file*

-d, --dev Enable development mode, which deactivates client side caching

-v, --verbose Enable basic debugging

-l LOG_LEVEL, --log-level=LOG_LEVEL Specify the log level: DEBUG, INFO, WARNING, ERROR, CRITICAL

-u LOGIN, --user=LOGIN Specify the login user

-s SERVER, --server=SERVER Specify the server hostname:port

1.2.4 URL

When an url is passed, the client will try to find already running client that could handle it and send to this one to open the url. If it doesn't find one then it will start the GUI and open the url itself.

The url schemes are:

tryton://<hostname>[:<port>]/<database>/model/<model name>[/<id>][;parameters]

tryton://<hostname>[:<port>]/<database>/wizard/<wizard name>[;parameters]

tryton://<hostname>[:<port>]/<database>/report/<report name>[;parameters]

where *parameters* are the corresponding fields of actions encoded in JSON.

Note: *model* is for *act_window*

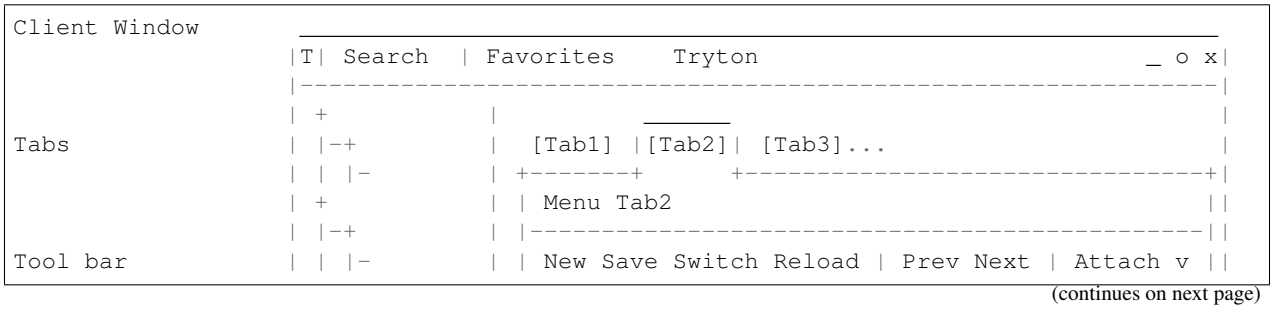
Note: *report* must have at least a data parameter with *ids*, *id* and *model name*

1.2.5 Overview

On startup the login dialog is displayed. It allows to select a existing profile (or to manage them) or to enter the host and database information.

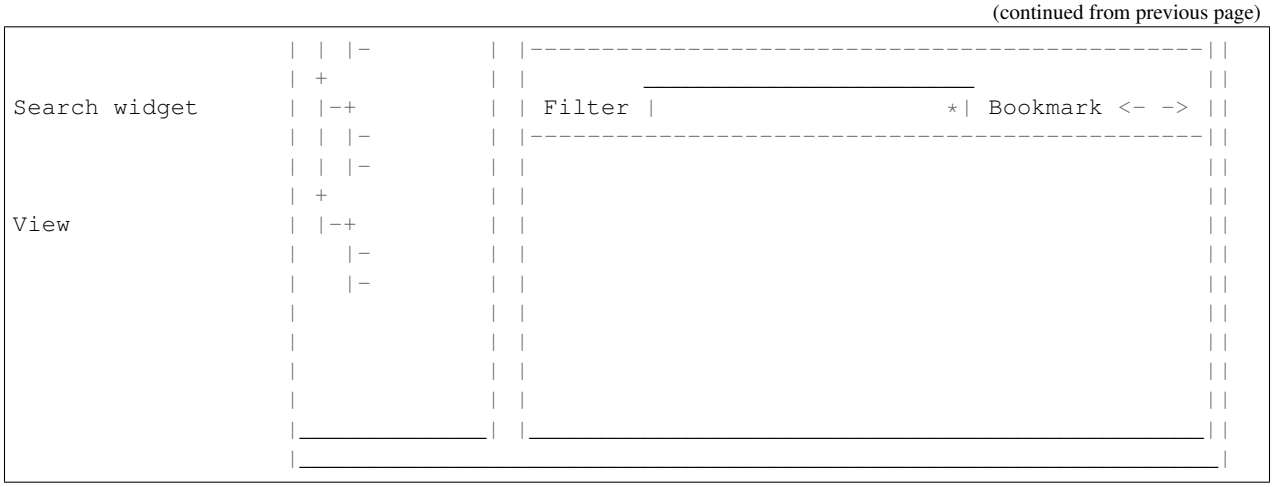
The following schematic illustration of the Tryton client shows the names of all important visual parts.

Figure: Tryton client application:



(continues on next page)

(continued from previous page)



Tabbed Main Frame

This part of the client contains all the related contents and functions provided by the *Tryton server modules*. All aspects inside the *main frame* depend at least on the individual set of installed modules.

The main frame provides a tabbed document interface to arrange different views side by side. New *tabs* are opened by special *actions*, like choosing a menu item or clicking some action buttons. All tabs include titles which show the name of the provided view.

Tabs can be arranged by Drag and Drop.

Note: Inside *views* there can be tabs, too.

Menu

The *menu* does not contain fixed menu items. All of them are dynamically provided by the actual set of the installed *modules* depending on the access rules of the current user. If a menu item is clicked, the appropriate action will open in a new tab.

A search field allows to quickly filter the menu items by name and to search in models for which the global search is enabled.

1.2.6 Application Menu

The following section describes the action of the application menu. A rule of thumb: All items of the menu bar that are suffixed by three dots (. . .) will open an intermediate *dialog* for setting up the provided menu action. Most dialog provide a *Cancel* button, used to stop the complete dialog process.

Preferences: A preference dialog opens, where the actual user can show and edit his personal settings. All user preferences are stored server side. I.e. logging in with the same credentials from different computers always restores the same preferences.

Options

The Options menu sets up several visual and context depending preferences.

Toolbar:

- Default: Shows labels and icons as defaulted in the GTK configuration.
- Text and Icons: Shows labels and icons in the tool bar.
- Icons: Shows icons only in the tool bar.
- Text: Shows labels only in the tool bar.

Form:

- Save Column Width: Check box to enable saving of manually adjusted widths of columns in lists and trees.
- Save Tree Expanded State: Check box to enable saving of expanded and selected nodes in trees/lists.
- Spell Checking: Check box to enable spell checking in fields.

PDA Mode: When activated, the client display in a condensed mode.

Search Limit: Open a dialog to set up the maximum number of records displayed on a list.

Email: Open a dialog to set up an email reader.

- Command Line: The command line calling the email reader.
- Placeholders:
  - `${to}`: the destination email address
  - `${cc}`: the carbon copy email address
  - `${subject}`: the subject of the email
  - `${body}`: the body of the email
  - `${attachment}`: the attachment of the email
- Examples:
  - Thunderbird 2 on Linux: `thunderbird -compose "to='${to}',cc='${cc}',subject='${subject}',body='${body}',attachment='file://${attachment}'"`
  - Thunderbird 2 on Windows XP SP3: `"C:\\Program Files\\Mozilla Thunderbird\\thunderbird.exe" -compose to="${to}",cc="${cc}",subject="${subject}",body="${body}",attachment="${attachment}"`

Note: The path of *Program Files* may vary dependent on the localization of your Windows version.

Check Version: Check box to enable the check of new bug-fix version.

Help

Keyboard Shortcuts...: Shows the information dialog of the predefined keyboard shortcut map.

- Edition Widgets: Shows shortcuts working on text entries, relation entries and date/time entries.

About...: License, Contributors, Authors of Tryton

1.2.7 Tool Bar

The tool bar contains the functionalities linked to the current tab. Some operations are working with one record or with a selection of *records*. In *form view* the actual record is selected for operations. In *tree view* all selected records are used for operations.

New: Creates a new record.

Save: Saves the actual record.

Switch View: Switches the actual view aspect to:

- *Form view*
- *Tree view*
- *Graph view*

Not all views provide all aspects.

Reload/Undo: Reloads the content of the actual tab. Undoes changes, if save request for the current record is denied.

Duplicate: Duplicates the content of the actual record in a newly created record.

Delete: Deletes the selected or actual record.

Previous: Goes to the last record in a list (sequence).

Next: Goes to the next record in a list (sequence).

Search: Goes to the search widget.

View Logs...: Shows generic information of the current record.

Show revisions...: Reload the current view/record at a specific revision.

Close Tab: Closes the current tab. A Request *Dialog* opens in case of unsaved changes.

Attachment: The attachment item handles the document management system of Tryton which is able to attach files to any arbitrary *model*. On click it opens the attachments *dialog*. The default dialog shows a list view of the attached files and links.

Actions...: Shows all actions for the actual view, model and record.

Relate...: Shows all relate view for the actual view, model and record.

Report...: Shows all reports for the actual view, model and record.

E-Mail...: Shows all email reports for the actual view, model and record.

Print...: Shows all print actions for the actual view, model and record.

Copy URL: Copy the URL of the form into the clipboard.

Export Data...: Export of current/selected records into *CSV*-file or open it in Excel.

- Predefined exports
  - Choose preferences of already saved exports.
- All Fields: Fields available from the model.
- Fields to export: Defines the specific fields to export.
- Options:
  - Save: Save export as a CSV file.
  - Open: Open export in spread sheet application.

- Add field names: Add a header row with field names to the export data.
- Actions:
  - Add: Adds selected fields to *Fields to export*.
  - Remove: Removes selected fields from *Fields to export*.
  - Clear: Removes all fields from *Fields to export*.
  - Save Export: Saves field mapping to a *Predefined export* with a name.
  - Delete Export: Deletes a selected *Predefined export*.
  - OK: Exports the data (action depending on *Options*).
  - Cancel

Import Data...: Import records from *CSV*-file.

- All Fields: Fields available in the model (required fields are marked up).
- Fields to Import: Exact sequence of all columns in the CSV file.
- File to Import: File *dialog* for choosing a CSV file to import.
- CSV Parameters: Setup specific parameters for chosen CSV file.
  - Field Separator: Character which separates CSV fields.
  - Text Delimiter: Character which encloses text in CSV.
  - Encoding: *Character encoding* of CSV file.
  - Lines to Skip: Count of lines to skip a headline or another offset.
- Actions:
  - Add: Adds fields to *Fields to Import*.
  - Remove: Deletes fields from *Fields to Import*.
  - Clear: Removes all fields from *Fields to Import*.
  - Auto-Detect: Tries to auto detect fields in the CSV *File to Import*.
  - OK: Proceeds the data import.
  - Cancel

1.2.8 Widgets

There are a several widgets used on Tryton in client side. The follow sections will explains some of them.

Date/DateTime/Time Widgets

These widgets have several key shortcuts to quickly modify the value. Each key increases if lower case or decreases if upper case:

- *y*: by one year
- *m*: by one month
- *w*: by one week
- *d*: by one day

- *h*: by one hour
- *i*: by one minute
- *s*: by one second

Search Widget

The search widget adds the ability to easily search for records on the current tab. This widget is visible only on *tree view*.

The Syntax

A query is composed of search clauses. A clause is composed of a field name (with *:* at the end), an operator and a value. The field name is optional and defaults to the record name. The operator is also optional and defaults to *like* or *equal* depending on the type of the field. The default operator is = except for fields of type *char*, *text* and *many2one* which is *ilike*.

Field Names

All field names shown in the *tree view* can be searched. Field names must be followed by a *:*

For example: `Name:`

If the field name contains spaces, it is possible to escape it using double quotes.

For example: `"Receivable Today":`

Operators

The following operators can be used:

- =: equal to
- <: less then
- <=: less then or equal to
- >: greater then
- >=: greater then or equal to
- *!*=: not equal
- *!*: not equal or not like (depending of the type of field)

For example: `Name: != Dwight`

Note: The *ilike* operator is never explicit and % is appended to the value to make it behaves like *starts with*

Values

The format of the value depends on the type of the field. A list of values can be set using *;* as separator.

For example: `Name:  Michael; Pam`

It will find all records having the *Name* starting with *Michael* or *Pam*.

A range of number values can be set using ...

For example: `Amount:  100..500`

It will find all records with *Amount* between *100* and *500* included.

There are two wildcards:

- %: matches any string of zero or more characters.
- _: matches any single character.

It is possible to escape special characters in values by using double quotes.

For example: `Name:  "Michael:Scott"`

Here it will search with the value *Michael:Scott*.

Clause composition

The clauses can be composed using the two boolean operators *and* and *or*. By default, there is an implicit *and* between each clause if no operator is specified.

For example: `Name:  Michael Amount:  100`

is the same as `Name:  Michael and Amount:  100`

The *and* operator has a highest precedence than *or* but you can change it by using parenthesis.

For example: `(Name:  Michael or Name:  Pam) and Amount:  100`

is different than `Name:  Michael or Name:  Pam and Amount:  100`

which is evaluated as `Name:  Michael or (Name:  Pam and Amount:  100)`

RichText Editor

This feature create a rich text editor with various features that allow for text formatting. The features are:

- Bold: On/off style of bold text
- Italic: On/off style of italic text
- Underline: On/off style of underline text
- Choose font family: Choice from a combo box the desired font family
- Choose font size: Choice from a combo box the desired size font
- Text justify: Choice between four options for alignment of the line (left, right, center, fill)
- Background color: Choose the background color of text from a color palette
- Foreground color: Choose the foreground color of text from a color palette

Besides these features, it can change and edit text markup. The text markup feature has a similar HTML tags and is used to describe the format specified by the user and is a way of storing this format for future opening of a correct formatted text. The tags are explain follows:

- Bold: Tag *b* is used, i.e. text
- Italic: Tag *i* is used, i.e. text
- Underline: Tag *u* is used, i.e. <u>text</u>
- Font family: It is a attribute *font-family* for *span* tag, i.e. text
- Font size: It is a attribute *size* for *span* tag, i.e. text
- Text Justify: For justification text is used paragraph tag *p*. The paragraph tag is used to create new lines and the alignment is applied across the board. Example: some text
- Background color: It is a attribute *background* for *span* tag, i.e. text
- Foreground color: It is a attribute *foreground* for *span* tag, i.e. text

1.2.9 CSS

The client can be styled using the file *theme.css*.

Here are the list of custom selectors:

- *.readonly*: readonly widget or label
- *.required*: widget or label of required field
- *.invalid*: widget for which the field value is not valid
- *headerbar.profile-<name>*: the name of the connection profile is set on the main window

For more information about style option see GTK+ CSS

1.2.10 Appendix

Configuration File

```
~/.config/tryton/x.y/tryton.conf      # General configuration
~/.config/tryton/x.y/accel.map        # Accelerators configuration
~/.config/tryton/x.y/known_hosts      # Fingerprints
~/.config/tryton/x.y/ca_certs         # Certification Authority (http://docs.python.
↪org/library/ssl.html#ssl-certificates)
~/.config/tryton/x.y/profiles.cfg     # Profile configuration
~/.config/tryton/x.y/plugins          # Local user plugins directory
~/.config.tryton/x.y/theme.css        # Custom CSS theme
```

Note: ~ means the home directory of the user. But on Windows system it is the *APPDATA* directory.

1.3 Glossary

Actions An *action* is a function which is triggered by a user intervention. *Actions* are called from activating menu items or pushing buttons. Actions often provide *wizards*.

Board The *board* is a type of *views* able to handle other views. This view type is not documented or not used for now.

Character Encoding See [WP-ENCOD]

CSV File format for Comma Separated Values. See [WP-CSV]

Data *Data* means information content produced by users.

Dialog A *dialog* is a *popup* window, which overlays other windows and request user interaction. *Dialogs* are used to set up special *actions*.

Fields *Fields* are attributes of a *data object*. *Fields* are represented as table fields in relational databases.

Form The *form* is the general type of *views* used in Tryton. The *form* provides several modes for presenting *data*:

- *Form View*
- *Tree View*
- *Graph View*

Form View The *form* is a mode of *views*, which displays single *records* of data.

Graph View *Graph view* is a mode of *views* to show sets of data in a diagram. *Graph views* can be pie-charts or bar-charts.

Main Frame The *main frame* is a huge part arranged in the center of the *Tryton client*. *Using the Tryton client* means mainly using the *main frame* part. It contains *tabs* to organize and to show different *views*.

Model A *model* describes how data is represented and accessed. Models formally define records and relationships for a certain domain of interest.

Modules *Modules* are enclosed file packages for the *Tryton server*. A *Module* defines the *Model*, the presentation of the information (*views*), functions, *actions* and default presets. Additionally *modules* may provide standardized data like ISO names for countries. *Modules* in Tryton are build up generically. That is, they are constructed as simple as possible to provide the desired functionality.

Plugins A *plugin* is an add-on module for the *Tryton client*.

Popup A small window which pops up the main window.

Records A *record* is a singular dataset in a *Model*. *Records* are represented as lines or *records* in a relational database table.

Tabs *Tabs* are *widgets* to arrange different contents side by side. They are used to switch quickly between different domains of interest. Tryton uses *tabs* in two layer:

- A tabbed *Main Frame*.
- Tabs inside *Views*.

The main frame consists of *tabs* that embed the main menu and all views to an appropriate *model*. The other type of *tabs* is used inside of *views* to split them into visual domains of the same model. These *tabs* are used for structuring contents of one model to different sub headings.

Three-Tiers A *three-tiers* application framework like Tryton, is build up of three different software components:

1. The storage or data tier.
2. The logic or application tier.
3. The presentation tier.

The storage tier in the Tryton framework is provided by the PostgreSQL database engine. The application logic tier is provided by *Tryton server* and its *modules*. The presentation tier is mainly provided by the *Tryton client*. In a *three tiers* framework, the presentation tier (client) never connects directly to the storage tier. All communication is controlled by the application tier.

Tree View *Tree view* is a mode of *views* showing sets of *data*. *Tree views* can be flat lists or tables as well as tree-like nested lists.

Tryton Client The *Tryton client* application is the graphical user interface (GUI) of the *Tryton server*.

Tryton Server The *Tryton server* is the application or logic tier in the *three-tiers* application platform *Tryton*. The *Tryton server* connects the underlying application logic of the different *modules* with corresponding database records. The *Tryton server* provides different interfaces to present the generated information:

- *Tryton client*: (graphical user interface GUI)
- XMLRPC see [WP-XMLRPC]
- WebDAV see [WP-WebDAV]
- OpenOffice

Views A *view* is the visual presentation of *data*. *Views* resides inside *tabs* in the *main frame* of the *Tryton client*. There are two general types of *views* in Tryton:

1. *Form*
2. *Board*

Each of the view types has different modes to show data. *Views* are built of several *widgets* and provide often additional *actions*. It is also possible to present the same data in different view modes alternately.

Widgets A *Widget* is a visual element of a graphical user interface (GUI). Some *Widgets* solely show information, others allow manipulation from user side. Example *Widgets* are buttons, check-boxes, entry-boxes, selection lists, tables, lists, trees, ...

Wizards *Wizards* define stateful sequences of interaction to proceed complex *actions*. A *wizard* divides the complexity of some actions into several user guided steps.

1.3.1 References

Bibliography

[WP-XMLRPC] http://en.wikipedia.org/wiki/Xmlrpc

[WP-WebDAV] http://en.wikipedia.org/wiki/Webdav

[WP-CSV] http://en.wikipedia.org/wiki/Comma-separated_values

[WP-ENCOD] http://en.wikipedia.org/wiki/Character_encoding

Index