Ledger Mode

Emacs Support For Version 3.0 of Ledger

Craig Earls

Table of Contents

1 Copying

2 Introduction to Ledger Mode

2.1 Quick Installation

The Emacs lisp source for Ledger-mode is included with the source distribution of Ledger. It is entirely included in the 'lisp' subdirectory. To use ledger mode include the following in your Emacs initialization file ('~/.emacs', '~/.emacs.d/init.el', '~/.Aquamacs/Preferences.el'

```
(add-to-list 'load-path (expand-file-name "/path/to/ledger/source/lisp/"))
(load "ldg-new")
(add-to-list 'auto-mode-alist '("\\.ledger$" . ledger-mode))
```

This sets up Emacs to automatically recognize files that end with '.ledger' and start Ledger mode. Nothing else should be required as long as the ledger command line utility is properly installed.

2.2 Menus

The vast majority of Ledger-mode functionality is available from the Emacs menu system. The keystrokes are shown in the menu to help you learn the faster keyboard methods.

2.3 Quick Demo

Load the demo file 'demo.ledger' from the Ledger source 'test/input' directory. The ledger will be loaded and font highlighted. At this point you could manually edit transactions and run Ledger from a convenient command line.

2.3.1 Quick Add

As simple as the ledger transaction format is, it can still be daunting to add many transactions manually. Ledger provides two way to add transactions with minimal typing. Both are based on the idea that most transactions are repetitions of earlier transactions.

In the 'demo.ledger' buffer enter a date using the correct format. Then type the first few characters of another payee in the 'demo.ledger' buffer. Type C-c TAB. Ledger-mode will search for a Payee that has the same beginning and copy the rest of the transaction to you new entry.

Additionally you can use the ledger xact command, by typing C-c C-a then typing a close match to the payee. Ledger mode will call ledger xact with the data you enter and place the transaction in the proper chronological place in the ledger.

2.3.2 Reconciliation

The biggest task of maintaining a ledger is ensuring the it matches the outside world. This process is called reconciliation (see Section 4.1 [Basics of Reconciliation], page 7) and can be quite onerous. Ledger mode attempts to make it as painless as possible.

In the 'demo.ledger' buffer type C-c C-r. Emacs will prompt for an account to reconcile in the mini-buffer. Enter Checking. Emacs will then prompt for a target value. The target value is the amount you want the cleared transactions in the buffer to total. Normally this would be the ending value from your bank statement, or the latest value in your on-line

transaction summary. Enter `1710`. Note that Ledger-mode assumes your are using $ (USD) as your default commodity, this can be easily changed in the customization variables. See Section 6.1 [Ledger-mode Customization], page 11.

You now see a list of uncleared transactions in a buffer below the 'demo.ledger' buffer. Touching the space bar will mark a transaction as pending and display the current cleared (and pending) balance, along with the difference remaining to meet your target. Clear the first three transactions, and you will see the difference to target reach $0. End the reconciliation by typing `C-c C-c`. This saves the demo.ledger buffer and marks the transactions and finally cleared. Type `q` to close out the reconciliation buffer.

2.3.3 Reports

The real power of Ledger is in it reporting capabilities. Reports can be run and displayed in a separate Emacs buffer. In the 'demo.ledger' buffer, type `C-c C-o C-r`. In the mini-buffer Emacs will prompt for a report name. There are a few built-in reports, and you can add any report you need See Section 5.2 [Adding and Editing Reports], page 9.

In the mini-buffer type `account`. When prompted for an account type `checking`. In another buffer you will see a Ledger register report. You can move around the buffer, with the point on a transaction, type `C-c C-c`. Ledger mode will take you directly to that transaction in the 'demo.ledger' buffer.

Another built-in report is the balance report. In the 'demo.ledger' buffer, type `C-c C-o C-r`. When prompted for a report to run, type `bal`, and a balance report of all accounts will be shown.

2.3.4 Folding

A ledger file can get very large. It can be helpful to collapse the buffer to display only the transactions you are interested in. Ledger-mode copies the `occur` mode functionality. Typing `C-c C-f` and entering any regex in the mini-buffer will show only transactions that match the regex. The regex can be on any field, or amount.

3 The Ledger Buffer

3.1 Adding Transactions

Beyond the two ways of quickly adding transactions (see Section 2.3.1 [Quick Add], page 2) Ledger-mode assists you by providing robust `TAB` completion for payees and accounts. Ledger-mode will scan the existing buffer for payees and accounts. Included files are not currently included in the completion scan. Repeatedly hitting `TAB` will cycle through the possible completions.

Ledger mode can also help you keep your amounts in alignment. Setting `ledger-post-auto-adjust-amounts` to true tells Ledger-mode to automatically place any amounts such that their last digit is aligned to the column specified by `ledger-post-amount-alignment-column`, which defaults to 52. See Section 6.2.5 [Ledger Post Customization Group], page 13.

3.2 Editing Amounts

GNU Calc is a very powerful Reverse Polish Notation calculator built into all recent version of Emacs. Ledger-mode makes it easy to calculate values for amount by integrating GNU Calc. With the point anywhere in the same line as a posting, typing `C-c C-b` will bring up the Calc buffer, and push the current amount for the posting onto the top of the Calc stack. Perform any calculations you need to arrive at the final value, then type `y` to yank the value at the top of stack back into the ledger buffer. Note: GNU Calc does not directly support commas as decimal separators. Ledger mode will translate values from decimal-comma format to decimal-period format for use in Calc, but it cannot intercept the value being yanked form the Calc stack, so decimal-comma users will have to manually replace the period with a comma.

3.3 Marking Transactions

Ledger considers transaction or posting to be in one of three states: uncleared, cleared, and pending. For calculation Ledger ignores these states unless specifically instructed to use them. Ledger-mode assigns some additional meaning to the states:

- Uncleared. No state. This is equivalent to sticking a check in the mail. It has been obligated, but not been cashed by the recipient. It could also apply to credit/debit card transactions that have not been cleared into your account balance. You bank may call these transactions 'pending', but Ledger-mode uses a slightly different meaning.

- Pending. Ledger-mode's reconciliation function see pending transactions as an intermediate step in reconciling an account. When doing a reconciliation (see Section 2.3.2 [Reconciliation], page 2), marking a transaction as pending means that you have seen the transaction finally recorded by the recipient, but you have not completely reconciled the account.

- Cleared. The transaction has been completely recognized by all parties to the transaction.

Clearing complete transactions is done by typing `C-c C-e` with point in a transaction. This places an asterisk (*) after the date. Clearing individual postings is done by typing `C-c C-c` while in a posting. This places an asterisk prior to the posting.

3.4 Deleting Transactions

Along with normal buffer editing methods to delete text, Ledger mode provides an easy way to delete the transaction under point: `C-c C-d`. The advantage to using this method is that the complete transaction operation is in the undo buffer.

3.5 Sorting Transactions

As you operating on the Ledger files, they may become disorganized. For the most part, Ledger doesn't care, but our human brains prefer a bit of order. Sorting the transactions in a buffer into chronological order can help bring order to chaos. Ledger sort (`C-c C-s`) will sort all of the transactions in a region by date. Ledger-mode isn't particularly smart about handling dates and it simply sorts the transactions using the string at the beginning of the transaction. So, you should use the preferred ISO 8601 standard date format `YYYY/MM/DD` which easily sorts.

Note, there is a menu entry to sort the entire buffer. Special transactions like automated transaction, will be moved in the sorting process and may not function correctly afterwards. For this reason there is no key sequence.

You can limit the allowed sort region by using embedded Ledger-mode markup within your ledger. For exmaple

```
<<< infomration to not sort >>>

; Ledger-mode: Start sort

<<< xacts to sort >>>

;Ledger-mode: End sort

<<< information to not sort >>>
```

You can use menu entries to insert start and end markers. These functions will automatically delete old markers and put new new marker at point.

3.6 Hiding Transactions

Often you will want to run Ledger register reports just to look at ax specific set of transactions. If you don't need the running total calculation handled by Ledger, Ledger-mode provides a rapid way of narrowing what is displayed in the buffer in a way that is simpler than the Ledger register command.

Based on the Emacs Occur mode by Alexey Veretennikov, Ledger-occur hides all transactions that do NOT meet a specific regular expression. The regular expression can match on any part of the transaction. If you want to find all transactions whose amount ends in .37, you can do that (I don't know why, but hey, whatever ever floats you aerostat).

Using `C-c C-f` or the `Hide Xacts` menu entry, enter a regular expression in the minibuffer. Ledger-mode will hide all other transactions. For details of the regular expression syntax, see ⟨undefined⟩ [(emacs)Regexps], page ⟨undefined⟩ or ⟨undefined⟩ [(elisp)Regular Expressions], page ⟨undefined⟩. A few examples using the 'demo.ledger' are given here:

'`Groceries`'

        Show only transactions that have a posting to the 'Groceries' account.

'^2011/01'
:   Show only transactions occurring in January of 2011.

'^2011/.*/25'
:   Show only transactions occurring on the 25th of the month in 2011

'.*ore'   Show only transaction with payees or accounts ending in 'ore'

To show all transactions simply invoke `Hide Xacts` or `C-c C-f` again.

4 The Reconcile Buffer

4.1 Basics of Reconciliation

Even in this relatively modern era, financial transactions do not happen instantaneously, unless you are paying cash. When you swipe your debit card the money may take several days to actually come out of your account, or a check may take several days to "clear". That is the root of the difference between "obligating" funds and "expending" funds. Obligation says you have agreed to pay it, the expenditure doesn't happen until the money actually leaves your account. Or in the case of receiving payment, you have an account receivable until the money has actually made it to you.

After an account has been reconciled you have verified that all the transactions in that account have been correctly recorded and all parties agree.

4.2 Starting a Reconciliation

To start reconciling an account you must have a target, both the transactions that you know about and the transactions the bank knows about. You can get this from a monthly statement, or from checking your online transaction history. It also helps immensely to know the final cleared balance you are aiming for.

Use menu `Reconcile Account` or `C-c C-r` and enter the account you wish to reconcile in the mini-buffer. Ledger-mode is not particular about what you enter for the account. You can leave it blank and Reconcile Mode will show you ALL uncleared transactions. After you enter the account enter the target amount. Ledger expects you to enter an amount with a commodity. It assumes initially that you are using $ (USD) as your default commodity. If you are working in a difference currency you can change the default in variable `ledger-reconcile-default-commodity` to whatever you need. If you work in multiple commodities simply enter the commoditized amount (for example `340 VSDX`, for 340 shares of VSDX).

Ledger-mode reconcile cannot currently reconcile accounts the have multiple commodities, such as brokerage accounts. You may use reconciliation mode to clear transactions, but balance calculations will not display the complete list of commodities.

4.3 Mark Transactions Pending

The '`*Reconcile*`' buffer will show all the uncleared transactions that meeting the criteria set in the regex. By default uncleared transactions are shown in red. When you have verified that a transaction has been correctly and completely recorded by the opposing party, mark the transaction as pending using the space bar. Continue this process until you agree with the opposing party and the difference from your target is zero.

4.4 Edit Transactions during Reconciliation

If you find errors during reconciliation. You can visit the transaction under point in the '`*Reconcile*`' buffer by hitting the `enter` key. This will take you to the transaction in the Ledger buffer. When you have finished editing the transaction saving the buffer will automatically return you to the '`*Reconcile*`' buffer and you can mark the transaction if appropriate.

4.5 Finalize Reconciliation

Once you have marked all transactions as pending and the cleared balance is correct. Finish the reconciliation by typing `C-c C-c`. This marks all pending transaction as cleared and saves the ledger buffer.

4.6 Adding and Deleting Transactions during Reconciliation

While reconciling, you may find new transactions that need to be entered into your ledger. Simply type `a` to bring up the quick add for the ledger buffer.

Typing `d` will delete the transaction under point in the '`*Reconcile*`' buffer form the ledger buffer.

4.7 Changing Reconciliation Account

You can conveniently switch the account being reconciled by typing `g`, and entering a new account to reconcile. This simply restarts the reconcile process. Any transactions that were marked 'pending' in the ledger buffer are left in that state when the account is switched.

4.8 Changing Reconciliation Target

If for some reason during reconciliation your target amount changes, type `t` and enter the new target value.

5 The Report Buffer

5.1 Running Reports

The real power behind Ledger is in its amazing reporting capability. Ledger-mode provides easy facility to run reports directly from Emacs. It has four reports built-in and facilities for adding custom reports.

Typing `C-c C-o C-r` or using menu `Ledger Run Report` prompt for the name of a saved report. The built-in reports are:

'`bal`'   Produce a balance reports of all accounts.

'`reg`'   Produce a register report of all transactions.

'`payee`'   Prompt for a payee, the produce a register report of all transaction involving that payee.

'`account`'   Prompt for an account, the produce a register report of all transaction involving that account.

5.2 Adding and Editing Reports

If you type a report name that Ledger-mode doesn't recognize it will prompt you for a ledger command line to run. That command is automatically saved with the name given and you can re-run it at any time.

There are two ways to edit the command line for a report. The first is to provide a prefix argument to the run-report command. For example, type `M-1 C-c C-o C-r`. This will prompt you for the report name, then present the report command line to be edited. When you hit enter, the report will be run, but it will not be permanently saved. If you want to save it, type `S` in the the '`*Ledger Report*`' buffer you will have the option to give it a new name, or overwrite the old report.

Deleting reports is accomplished by type `C-c C-o C-e` Edit Reports in the ledger buffer, or typing `E` in the '`*Ledger Report*`' buffer. This takes you to the Emacs customization window for the `ledger-reports` variable. Use the widgets to delete the report you want removed.

5.2.1 Expansion Formats

It is sometime convenient to leave room to customize a report without saving the command line every time. For example running a register report for a specific account, enter at runtime by the user. The built-in report '`account`' does exactly that, using a variable expansion to prompt the user for the account to use. There are four variable that can be expanded to run a report:

'`ledger-file`'
                Returns the file to be operated on.

'`payee`'   Prompts for a payee.

'`account`'   Prompt for an account.

'`value`'   Prompt for a tag value.

You can use these expansion values in your ledger report commands. For example, if you wanted to specify a register report the displayed transactions from a user-determined account with a particular meta-data tag value, you specify the following command line:

```
ledger -f %(ledger-file) reg %(account)  --limit \"tag('my-tag') =~
/%(value)/\"
```

Note how the double-quotes are escaped with back-slashes.

5.2.2 Make Report Transactions Active

In a large register report it is convenient to be able to jump to the source transaction. Ledger-mode will automatically include source information in every register file that doesn't contain a `--subtotal` option. It does this by adding a `--prepend-format='%(filename):%(beg_line):'` to the register report command-line you specify. You should never have to see this, but if there is an error in your ledger output this additional information may not get stripped out of the visible report.

5.3 Reversing Report Order

Often, banks show their online transaction histories with the most recent transaction at the top. Ledger itself cannot do a sensible ledger report in reverse chronological order, if you sort on reverse date the calculation will also run in the opposite direction. If you want to compare a ledger register report to a bank report with the most recent transactions at the top, type R in the '`*Ledger Report*`' buffer and it will reverse the order of the transactions and maintain the proper mathematical sense.

6 Customizing Ledger-mode

6.1 Ledger-mode Customization

Ledger-mode has several options available for configuration. All options can be configure through the Emacs customization menus, or specified in your Emacs initialization file. The complete list of options is show below. To change the option using the Emacs customization menu, simply chose customize in the Options menu and look for Ledger under the data options. Alternately you can choose "Customize Specific Group" and enter "Ledger" as the group.

6.2 Customization Variables

6.2.1 Ledger Customization Group

`ledger-occur-use-face-unfolded`
> If non-nil use a custom face for xacts shown in 'ledger-occur' mode using `ledger-occur-xact-face`.

`ledger-clear-whole-transactions`
> If non-nil, clear whole transactions, not individual postings.

`ledger-highlight-xact-under-point`
> If non-nil highlight xact under point using `ledger-font-highlight-face`.

6.2.2 Ledger Reconcile Customization Group

`ledger-reconcile-default-commodity`
> The default commodity for use in target calculations in ledger reconcile. Defaults to $ (USD)

`ledger-recon-buffer-name`
> Name to use for reconciliation window.

`ledger-fold-on-reconcile`
> If non-nil, limit transactions shown in main buffer to those matching the reconcile regex.

`ledger-buffer-tracks-reconcile-buffer`
> If non-nil, then when the cursor is moved to a new xact in the recon window.

`ledger-reconcile-force-window-bottom`
> If non-nil, make the reconcile window appear along the bottom of the register window and resize.

`ledger-reconcile-toggle-to-pending`
> If non-nil, then toggle between uncleared and pending (`!`). If false toggle between uncleared and cleared (`*`)

6.2.3 Ledger Report Customization Group

`ledger-reports`
> Definition of reports to run.

`ledger-report-format-specifiers`
> An alist mapping ledger report format specifiers to implementing functions.

6.2.4 Ledger Faces Customization Group

Ledger Faces : Ledger mode highlighting

`ledger-font-uncleared-face`
> Default face for Ledger

`ledger-font-cleared-face`
> Default face for cleared (*) transactions

`ledger-font-highlight-face`
> Default face for transaction under point

`ledger-font-pending-face`
> Default face for pending (!) transactions

`ledger-font-other-face`
> Default face for other transactions

`ledger-font-posting-account-face`
> Face for Ledger accounts

`ledger-font-posting-amount-face`
> Face for Ledger amounts

`ledger-occur-folded-face`
> Default face for Ledger occur mode hidden transactions

`ledger-occur-xact-face`
> Default face for Ledger occur mode shown transactions

`ledger-font-comment-face`
> Face for Ledger comments

`ledger-font-reconciler-uncleared-face`
> Default face for uncleared transactions in the reconcile window

`ledger-font-reconciler-cleared-face`
> Default face for cleared (*) transactions in the reconcile window

`ledger-font-reconciler-pending-face`
> Default face for pending (!) transactions in the reconcile window

`ledger-font-report-clickable-face`
> Default face for pending (!) transactions in the reconcile window

6.2.5 Ledger Post Customization Group

Ledger Post :

`ledger-post-auto-adjust-amounts`
> If non-nil, then automatically align amounts to column specified in `ledger-post-amount-alignment-column`

`ledger-post-amount-alignment-column`
> The column Ledger-mode uses to align amounts

`ledger-default-acct-transaction-indent`
> Default indentation for account transactions in an entry.

`ledger-post-use-completion-engine`
> Which completion engine to use, iswitchb, ido, or built-in

`ledger-post-use-ido`

6.2.6 Ledger Exec Customization Group

Ledger Exec : Interface to the Ledger command-line accounting program.

`ledger-binary-path`
> Path to the ledger executable.

`ledger-init-file-name`
> Location of the ledger initialization file. nil if you don't have one

6.2.7 Ledger Test Customization Group

`ledger-source-directory`
> Directory where the Ledger sources are located.

`ledger-test-binary`
> Directory where the debug binary.

6.2.8 Ledger Texi Customization Group

`ledger-texi-sample-doc-path`
> Location for sample data to be used in texi tests, defaults to '`~/ledger/doc/sample.dat`'

`ledger-texi-normalization-args`
> texi normalization for producing ledger output, defaults to "`--args-only --columns 80`"

7 Generating Ledger Regression Tests

Work in Progress.

8 Embedding Example results in Ledger Documentation

Work in Progress.

9 Hacking Ledger-mode

Work in Progress.