Ledger Mode

Emacs Support For Version 3.0 of Ledger

Craig Earls

Table of Contents

1 Introduction to Ledger-mode

1.1 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`, or `~/.Aquamacs/Preferences.el`).

```
(autoload 'ledger-mode "ledger-mode" "A major mode for Ledger" t)
(add-to-list 'load-path
             (expand-file-name "/path/to/ledger/source/lisp/"))
(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.

1.2 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.

1.3 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.

1.3.1 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 either typing *`C-c C-a`* or using '`Add Transaction`' menu entry. 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.

If you need to add a lot of transactions that are not near your current date you can set the current year and month so that using '`Add Transaction`' will prompt you with a more convenient month and year. To set the month type *`C-c RET`* and enter the month you want. *`C-c C-y`* will prompt you for the year. These settings only effect the '`Add Transaction`' command.

1.3.2 Reconciliation

The biggest task of maintaining a ledger is ensuring that it matches the outside world. This process is called reconciliation (see Section 3.1 [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*. If cursor is on an account, Ledger-mode will propose this account, or in the Minibuffer, will prompt for an account to reconcile. Hit *RET* if you are happy with proposed account, or enter '`Checking`' as example. 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 12.

You now see a list of uncleared transactions in a buffer below the `demo.ledger` buffer. Touching the *SPC* 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.

1.3.3 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 Minibuffer Emacs will prompt for a report name. There are a few built-in reports, and you can add any report you need See Section 4.2 [Adding and Editing Reports], page 9.

In the Minibuffer type '`account`'. When prompted for an account type '`checking`'. In a buffer named `*Ledger Report*`, you will see a Ledger register report. You can move around the buffer, with the point on a transaction, type *RET*. 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.

1.3.4 Narrowing

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 Minibuffer will show only transactions that match the regex. The regex can be on any field, or amount. Use *C-c C-g* after editing transactions to re-apply the current regex. Cancel the narrowing by typing *C-c C-f* again.

2 The Ledger Buffer

2.1 Adding Transactions

Beyond the two ways of quickly adding transactions (see Section 1.3.1 [Quick Add], page 1) 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 aligned. 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 14.

2.1.1 Setting a Transactions Effective Date

Ledger provides for adding information to a transaction that add details to the dates. For example, you can specify when the transaction was entered, when the transaction was cleared, or when individual postings were cleared. Ledger-mode refers to these additional dates as *effective* dates. To set the effective date of a transaction, place the point in the first line of a transaction and type *C-c C-t*. The effective date will be added to the transaction. To set the effective date for an individual posting, place point in the posting and type *C-c C-t* and the effective date for that posting will be added at the end of the posting.

2.1.2 Quick Balance Display

You will often want to quickly check the balance of an account. The easiest way it to position point on the account you are interested in, and type *C-c C-p*. The Minibuffer will ask you to verify the name of the account you want, if it is already correct hit *RET*, then the balance of the account will be displayed in the Minibuffer.

2.2 Copying Transactions

An easy way to copy a transaction is to type *C-c C-k* or menu entry '`Copy Trans at Point`'. You will be prompted the new date for the copied transaction, and after having confirmed with *RET*, new transaction will be inserted at *date* position in buffer.

2.3 Editing Amounts

GNU Emacs Calculator, aka '`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 `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: `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.

2.4 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 1.3.2 [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.

Typing *C-c C-c*, depending where is the point, will clear the complete transaction, or an individual posting. This places an asterisk '`*`' prior to the payee for the complete transaction, or prior to the account for an individual posting. When point is inside a transaction, specifically on an individual posting, you can still clear the complete transaction by typing *C-c C-e*.

2.5 Formatting Transactions

When editing a transaction, liberal use of the *TAB* key can keep the transaction well formatted. If you want to have Ledger-mode cleanup the formatting of a transaction you can use '`Align Transaction`' or '`Align Region`' from the menu bar.

The menu item '`Clean-up Buffer`' sorts all transactions in the buffer by date, removes extraneous empty lines and aligns every transaction.

2.6 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.

2.7 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. Either using '`Sort Region`' menu entry or typing *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 '`Sort Buffer`' 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 example:

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

; Ledger-mode: Start sort

<<< transactions to sort >>>

; Ledger-mode: End sort

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

You can use menu entries '`Mark Sort Beginning`' to insert start and '`Mark Sort End`' to insert end markers. These functions will automatically delete old markers and put new new marker at point.

2.8 Narrowing Transactions

Often you will want to run Ledger register reports just to look at a 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 '`Narrow to Regex`' menu entry, enter a regular expression in the Minibuffer. Ledger-mode will hide all other transactions. For details of the regular expression syntax, see your Emacs documentation. 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.

'`auto`' Show only transactions with payees or accounts or comments containing. '`auto`'

'`harley$`' Show only transactions with any line ending with '`harley`'.

To show back all transactions simply invoke '`Narrow to Regex`' or *`C-c C-f`* again.

If you've edited some transactions after narrowing such that they would no longer match the regular expression, you can refresh the narrowed view using `C-c C-g`.

3 The Reconcile Buffer

3.1 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.

3.2 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 on-line transaction history. It also helps immensely to know the final cleared balance you are aiming for.

Use menu '`Reconcile Account`' or keyboard shortcut *`C-c C-r`* to start reconciliation.

If cursor is on an account, Ledger-mode will propose this account, or in the Minibuffer, will prompt for an account to reconcile. Hit *`RET`* if you are happy with proposed account, or enter '`Checking`' as example. Ledger-mode is not particular about what you enter for the account. You can leave it blank and `*Reconcile*` buffer will show you *all* uncleared transactions.

After you enter the account enter the target amount. It is helpful to enter an amount with a commodity. You can also leave it blank, you will be able to clear transactions but not benefit from balance calculations. It assumes initially that you are using '`$`' (USD) as your default commodity. If you are working in a different 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 that 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.

3.3 Mark Transactions Pending

The `*Reconcile*` buffer will show all the uncleared transactions that meet 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 *`SPC`* bar. Continue this process until you agree with the opposing party and the difference from your target is zero.

3.4 Edit Transactions during Reconciliation

If you find errors during reconciliation. You can visit the transaction under point in the `*Reconcile*` buffer by hitting the *RET* 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.

3.5 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 transactions as cleared and saves the ledger buffer.

Type *q* to close out the reconciliation buffer. If variable *ledger-reconcile-finish-force-quit* is set, the reconciliation buffer will be killed automatically after *C-c C-c*.

3.6 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 from the ledger buffer.

3.7 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.

3.8 Changing Reconciliation Target

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

4 The Report Buffer

4.1 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 '`Run Report`' prompts 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, then produce a register report of all transactions involving that payee.

account Prompt for an account, then produce a register report of all transactions involving that account.

While viewing reports you can easily switch back and forth between the ledger buffer and the `*Ledger Report*` buffer. In `*Ledger Report*` buffer, typing `RET` will take you to that transaction in the ledger buffer. While in the ledger buffer `C-c C-o C-g` returns you to the `*Ledger Report*` buffer.

By default Ledger-mode will refresh the report buffer when the ledger buffer is saved. If you want to rerun the report at another time `C-c C-o C-a`. This is useful if you have other programs altering your ledger file outside of Emacs.

4.2 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 `RET`, the report will be run, but it will not be permanently saved. If you want to save it, type `S` in 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 typing `C-c C-o C-e` or using '`Edit Report`' menu in the ledger buffer, or typing `e` in the `*Ledger Report*` buffer. This takes you to the Emacs customization window for the Ledger Reports variables. Use the widgets to delete the report you want removed.

Typing `C-c C-o C-s` will prompt for a name and save the current report.

4.2.1 Expansion Formats

It is sometimes 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 entered

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 variables 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.

tagname Prompt for a meta-data tag name.

tagvalue Prompt for a meta-data 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.

4.2.2 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 `--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.

4.3 Reversing Report Order

Often, banks show their on-line 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.

5 Scheduling Transactions

The Ledger program provides for automating transactions but these transaction aren't *real*, they only exist inside a ledger session and are not reflected in the actual data file. Many transactions are very repetitive, but may vary slightly in the date they occur on, or the amount. Some transactions are weekly, monthly, quarterly or annually. Ledger mode provides a way to schedule upcoming transaction with a flexible scheduler that allows you to specify the transactions in a separate ledger file and calculate the upcoming occurrences of those transactions. You can then copy the transactions into your live data file.

5.1 Specifying Upcoming Transactions

The format for specifying transactions is identical to Ledger's file format with the exception of the date field. The data field is modified by surrounding it with brackets and using wild cards and special characters to specify when the transactions should appear.

5.1.1 Transactions that occur on specific dates

Many times you will enter repetitive transactions that occur on the same day of the month each month. These can be specified using a wild card in the year and month with a fixed date in the day. The following entry specifies a transaction that occurs on the first and fifteenth of every month in every year.

```
[*/*/1,15] Paycheck
    Income:Job       $1000.00
    Assets:Checking
```

Some transactions do not occur every month. Comma separated lists of the months, or '`E`' for even, or '`O`' for odd number months can also be specified. The following entry specifies a bi-monthly exterminator bill that occurs in the even months:

```
[*/E/01]  Exterminator
    Expenses:Home   $100.00
    Assets:Checking
```

5.1.2 Transactions that occur on specific days

Some transactions occur every relative to the day of the week rather than the date of the month. For example, many people are paid every two weeks without regard to the day of the month. Other events may occur on specific days regardless of the date. For example the following transactions creates a transaction every other Thursday:

```
[2014/11/27+2Th]  Paycheck
    Income:Job       $1000.00
    Assets:Checking
```

It is necessary to specify a starting date in order for this type of recurrence relation to be specified. The day names are two character codes that default to Mo, Tu, We, Th, Fr, Sa, Su, for Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday respectively. You can change the codes to something more convenient for your locale by customizing the ledger `ledger-schedule-week-days`. They must be two characters long.

6 Customizing Ledger-mode

6.1 Ledger-mode Customization

Ledger-mode has several options available for configuration. All options can be configured through the Emacs customization menus, or specified in your Emacs initialization file. The complete list of options is shown 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-shown`
: If non-nil, use a custom face for transactions 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 transaction under point using `ledger-font-highlight-face`.

6.2.2 Ledger Reconcile Customization Group

`ledger-recon-buffer-name`
: Name to use for reconciliation buffer. Defaults to `*Reconcile*`.

`ledger-narrow-on-reconcile`
: If t, limit transactions shown in main buffer to those matching the reconcile regex.

`ledger-buffer-tracks-reconcile-buffer`
: If t, then when the cursor is moved to a new transaction in the `*Reconcile*` buffer. Then that transaction will be shown in its source buffer.

`ledger-reconcile-force-window-bottom`
: If t, make the `*Reconcile*` window appear along the bottom of the register window and resize.

`ledger-reconcile-toggle-to-pending`
: If t, then toggle between uncleared and pending '`!`'. If false toggle between uncleared and cleared '`*`'.

`ledger-reconcile-default-date-format`
: Date format for the reconcile buffer. Defaults to `ledger-default-date-format`.

`ledger-reconcile-target-prompt-string`
: Prompt for recon target. Defaults to "Target amount for reconciliation ".

`ledger-reconcile-buffer-header`
: Header string for the reconcile buffer. If non-nil, the name of the account being reconciled will be substituted into the '%s'. If nil, no header will be displayed. Defaults to "Reconciling account %s\n\n".

`ledger-reconcile-buffer-line-format`
: Format string for the ledger reconcile posting format. Available fields are date, status, code, payee, account, amount. The format for each field is %WIDTH(FIELD), WIDTH can be preceded by a minus sign which mean to left justify and pad the field. WIDTH is the minimum number of characters to display; if string is longer, it is not truncated unless `ledger-reconcile-buffer-payee-max-chars` or `ledger-reconcile-buffer-account-max-chars` is defined. Defaults to "%(date)s %-4(code)s %-50(payee)s %-30(account)s %15(amount)s\n"

`ledger-reconcile-buffer-payee-max-chars`
: If positive, truncate payee name right side to max number of characters.

`ledger-reconcile-buffer-account-max-chars`
: If positive, truncate account name left side to max number of characters.

`ledger-reconcile-sort-key`
: Key for sorting reconcile buffer. Possible values are '(date)', '(amount)', '(payee)' or '(0)' for no sorting, i.e. using ledger file order. Defaults to '(0)'.

`ledger-reconcile-insert-effective-date` `nil`
: If t, prompt for effective date when clearing transactions during reconciliation.

`ledger-reconcile-finish-force-quit` `nil`
: If t, will force closing reconcile window after `C-c C-c`.

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-account-cleared-face`
: Face for cleared Ledger accounts.

`ledger-font-posting-account-pending-face`
: Face for Ledger pending accounts.

`ledger-font-posting-amount-face`
: Face for Ledger amounts.

`ledger-occur-narrowed-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*` buffer.

`ledger-font-reconciler-cleared-face`
: Default face for cleared '`*`' transactions in the `*Reconcile*` buffer.

`ledger-font-reconciler-pending-face`
: Default face for pending '`!`' transactions in the `*Reconcile*` buffer.

`ledger-font-report-clickable-face`
: FIXME

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.

Concept Index

Command & Variable Index

Keystroke Index