NAME

ledger - Command-line, double-entry account reporting tool

SYNOPSIS

ledger [*options*] [*command*] [*arguments*]

DESCRIPTION

ledger is a command-line accounting tool based on the power and completeness of double-entry accounting. It is only a reporting tool, which means it never modifies your data files, but it does offer a large selection of reports, and different ways to customize them to your needs.

COMMANDS

ledger accepts several top-level commands, each of which generates a different kind of basic report. Most of them accept a *report-query* argument, in order to determine what should be reported. To understand the syntax of a *report-query*, see the section on *QUERIES*. In its most basic form, simply specifying one or more strings produces a report for all accounts containing those strings.

If no command is given, **ledger** enters a REPL, or command loop, allowing several commands to be executed on the same dataset without reparsing.

The following is a complete list of accepted reporting commands:

accounts [*report-query*]

List all accounts for postings that match the *report-query*.

balance [*report-query*]

Print a balance report showing totals for postings that match *report-query*, and aggregate totals for parents of those accounts. Options most commonly used with this command are:

--basis (**-B**) Report in terms of cost basis, not amount or value. This is the only form of report which is guaranteed to always balance to zero, when no *report-query* is specified. Only show totals for the top-most accounts.

--empty (**-E**) Show accounts whose total is zero.

--flat Rather than display a hierarchical tree, flatten the report to show subtotals for only accounts matching *report-query*.

--no-total Suppress the summary total shown at the bottom of the report.

The synonyms **bal** and **b** are also accepted.

budget [*report-query*]

A special balance report which includes three extra columns: the amount budgeted during the

reporting period, how spending differed from the budget, and the percentage of budget spent (exceeds 100% if you go over budget). Note that budgeting requires one or more "periodic transactions" to be defined in your data file(s). See the manual for more information.

cleared [*report-query*]
A special balance report which adds two extra columns: the cleared balance for each account, and the date of the most recent cleared posting in that account. For this accounting to be meaningful, the cleared flag must be set on at least one posting. See the manual for more information.

commodities [*report-query*]
List all commodities for postings matching the *report-query*.

convert Reads data from a CSV (comma-separated values) file and generates **ledger** transactions.

csv [*report-query*]
Report of postings matching the *report-query* in CSV format (comma-separated values). Useful for exporting data to a spreadsheet for further analysis or charting.

entry [*entry-template*]
Generate and display a new, properly formatted **ledger** transaction by comparing the *entry-template* to the transactions in your data file(s). For more information on draft templates and using this command to quickly create new transactions, see the section *ENTRIES*.

The synonym **xact** is also accepted.

emacs [*query*]
Output posting and transaction data in a format readily consumed by the Emacs editor, in a series of Lisp forms. This is used by the Emacs ledger-mode to process reporting data from **ledger**.

equity [*report-query*]
Print a transaction with a series of postings that balance current totals for accounts matching the *report-query* in a special account called Equity:Opening Balances. The purpose of this report is to close the books for a prior year, while using these equity postings to carry forward those balances.

payees [*report-query*]
List all payees for postings matching the *report-query*.

pricemap

Produce a file which can be used to generate a graph with graphviz showing the relationship of commodities in the **ledger** file.

prices [*report-query*]

Report prices for all commodities in postings matching the *report-query*. The prices are reported with the granularity of a single day.

pricedb [*report-query*]

Report prices for all commodities in postings matching the *report-query*. Prices are reported down to the second, using the same format as the *~/.pricedb* file.

print [*report-query*]

Print out the full transactions of any matching postings using the same format as they would appear in a data file. This can be used to extract subsets from a **ledger** file to transfer to other files.

push [*options*]

In the REPL, push a set of command-line *options*, so that they will apply to all subsequent reports.

pop In the REPL, pop any option settings that have been **push**ed.

register [*report-query*]

List all postings matching the *report-query*. This is one of the most common commands, and can be used to provide a variety of useful reports. Options most commonly used with this command are:

--average (**-A**)

Show the running average, rather than a running total.

--current (**-c**) Don't show postings beyond the present day.

--exchange "*COMMODITY* [, *COMMODITY, ...*]" (**-X**)

Render all values in the given *commodity*, if a price conversion rate can be determined. If multiple commodities are given, values in a listed commodity will remain as-is, and others will be displayed in the first listed commodity they can be converted to. Rates are always displayed relative to the date of the posting they are calculated for. This means a **register** report is a historical value report. For current values, it may be preferable to use the **balance** report.

--gain (**-G**) Show any gains (or losses) in commodity values over time.

--head *number*

Only show the top *number* postings.

--historical (**-H**)
Value commodities at the time of their acquisition.

--invert Invert the value of amounts shown.

--market (**-V**) Show current market values for all amounts. This is determined in a somewhat magical fashion. It is probably more straightforward to use **--exchange** option.

--period *time-period* (**-p**)
Show postings only for the given *time-period*.

--related (**-r**) Show postings that are related to those that would have been shown. It has the effect of displaying the "other side" of the postings.

--sort *value-expression* (**-S**)
Sort postings by evaluating the given *value-expression*. Note that a comma-separated list of expressions is allowed, in which case each sorting term is used in order to determine the final ordering. For example, to search by date and then amount, one would use:

ledger reg --sort 'date, amount'

The sort order may be controlled with the '-' sign. For example, to sort in reverse chronological order:

ledger reg --sort '-date'

--tail *number* Only show the last *number* postings.

--uncleared (**-U**)
Only show uncleared (i.e., recent) postings.

There are also several grouping options that can be useful:

--by-payee (**-P**)
Group postings by common payee names.

--daily (**-D**) Group postings by day.

--weekly (**-W**)
Group postings by week (starting on Sundays).

--start-of-week *day*
Set the start of each report grouped by week to the given *day*.

--monthly (**-M**)
Group postings by month.

--quarterly Group postings by fiscal quarter.

--yearly (**-Y**) Group postings by year.

--days-of-week
Group postings by the day of the week on which they took place.

--subtotal (**-s**) Group all postings together. This is very similar to the totals shown by the **balance** report.

The synonyms **reg** and **r** are also accepted.

select [*sql-query*]

List all postings matching the *sql-query*. This command allows to generate SQL-like queries, e.g.:

ledger select date,amount from posts where account=~/Income/

source Parse a journal file and checks it for errors. **ledger** will return success if no errors are found.

stats [*report-query*]

Provide summary information about all the postings matching *report-query*. It provides information such as:

- Time range of all matching postings
- Unique payees
- Unique accounts
- Postings total
- Uncleared postings
- Days since last posting
- Posts in the last 7 days
- Posts in the last 30 days
- Posts this month

xml [*report-query*]

Output data relating to the current report in XML format. It includes all accounts and commodities involved in the report, plus the postings and the transactions they are contained in. See the manual for more information.

OPTIONS

--abbrev-len *INT*

Set the minimum length an account can be abbreviated to if it doesn't fit inside the **account-width**. If *INT* is zero, then the account name will be truncated on the right. If *INT* is greater than **account-width** then the account will be truncated on the left, with no shortening of the account names in order to fit into the desired width.

--account *EXPR*

Prepend *EXPR* to all accounts reported. That is, the option **--account** *"'Personal'"* would tack *Personal:* and **--account** *"tag('VAT')"* would tack the value of the VAT tag to the beginning of every account reported in a **balance** or **register** report.

--account-width *INT*

Set the width of the account column in the **register** report to *INT* characters.

--actual (**-L**)
Report only real transactions, with no automated or virtual transactions used.

--add-budget
Show only un-budgeted postings.

--align-intervals
Use the beginning of the reporting period as the start of intervals, rather than the beginning of the week, month, quarter or year.

--amount *EXPR* (**-t**)
Apply the given value expression to the posting amount. Using **--amount** *EXPR* you can apply an arbitrary transformation to the postings.

--amount-data (**-j**)
On a register report print only the dates and amount of postings. Useful for graphing and spreadsheet applications.

--amount-width *INT*
Set the width in characters of the amount column in the **register** report.

--anon Anonymize registry output, mostly for sending in bug reports.

--ansi Use color if the terminal supports it. Alias for **--color**

--args-only
Ignore init files and environment variables for the **ledger** run.

--auto-match
When generating a ledger transaction from a CSV file using the **convert** command, automatically match an account from the Ledger journal.

--aux-date
Show auxiliary dates for all calculations. Alias for **--effective**

--average (**-A**)
Print average values over the number of transactions instead of running totals.

--average-lot-prices
Report the average price at which each commodity was purchased in a balance report.

--balance-format *FMT*
Specify the format to use for the **balance** report.

--base Reduce convertible commodities down the bottom of the conversion, e.g. display time in seconds.

--basis (**-B**)
Report the cost basis on all posting. Alias for **--cost**

--begin *DATE* (**-b**)
Specify the start *DATE* of all calculations. Transactions before that date will be ignored.

--bold-if *EXPR*
Print the entire line in bold if the given value expression is true.

--budget
Only display budgeted items. In a **register** report this displays transaction in the budget, in a balance report this displays accounts in the budget.

--budget-format *FMT*
Specify the format to use for the **budget** report.

--by-payee (**-P**)
Group postings in the register report by common payee names.

--check-payees
Enable strict and pedantic checking for payees as well as accounts, commodities and tags.

--cleared (**-C**)
Display only cleared postings.

--cleared-format *FMT*
Specify the format to use for the **cleared** report

--collapse (**-n**)
Print only the top level accounts.

--collapse-if-zero
Collapse the account display only if it has a zero balance.

--color Use color if the terminal supports it. Alias for **--ansi**

--columns *INT*
Make the **register** report *INT* characters wide. By default **ledger** will use all available columns in your terminal.

--cost Report the cost basis on all posting. Alias for **--basis**.

--count
Direct **ledger** to report the number of items when appended to the **commodities**, **accounts** or **payees** commands.

--csv-format *FMT*
Format **csv** report according to *FMT*.

--current (**-c**)
Shorthand for **--limit** *'date <= today'*.

--daily (**-D**)
Shorthand for **--period** *daily*.

--date *EXPR*
Transform the date of the transaction using *EXPR*.

--date-format *DATEFMT* (**-y**)
Print dates using *DATEFMT*. Refer to strftime(3) for details on the format string syntax.

--datetime-format *DATETIMEFMT*
Print datetimes using *DATETIMEFMT*. Refer to strftime(3) for details on the format string syntax.

--date-width *INT*
Specify the width, in characters, of the date column in the **register** report.

--day-break
Break up **register** report of timelog entries that span multiple days by day.

--days-of-week
Group transactions by the days of the week. Alias for **--dow**.

--dc Display register or balance in debit/credit format If you use **--dc** with either the **register** or **balance** commands, you will now get separate columns for debits and credits.

--debug *STR*
If **ledger** has been built with debug options this will provide extra data during the run.

--decimal-comma
Direct **ledger** to parse journals using the European standard comma as decimal separator, vice a period.

--depth *INT*
Limit the depth of displayed accounts in balance and register reports. Any accounts of greater depth are folded into their parent at the specified level. For example with **--depth** *2* the account **Expenses:Entertainment:Dining** would be folded into **Expenses:Entertainment** for display. Importantly, this is a display predicate, which means it only affects display, not the total calculations.

--detail
Related to **convert** command. Synonym to **--rich-data** option.

--deviation
Report each posting's deviation from the average. It is only meaningful in the **register** and **prices** reports.

--display *EXPR* (**-d**)
Display lines that satisfy the expression *EXPR*.

--display-amount *EXPR*
Apply a transformation to the *displayed* amount. This occurs after calculations occur.

--display-total *EXPR*
Apply a transformation to the *displayed* total. This occurs after calculations occur.

--dow Group transactions by the days of the week. Alias for **--days-of-week**.

--download
Cause quotes to be automagically downloaded, as needed, by running a script named *getquote*

and expecting that script to return a value understood by **ledger**. A sample implementation of a *getquote* script, implemented in Perl, is provided in the distribution. Downloaded quote price are then appended to the price database, usually specified using the environment variable LEDGER_PRICE_DB.

--effective
Show auxiliary dates for all calculations. Alias for **--aux-date**.

--empty (**-E**)
Include empty accounts in report.

--end *DATE* (**-e**)
Constrain the report so that transactions on or after *DATE* are not considered.

--equity
Related to the **equity** command. Gives current account balances in the form of a register report.

--exact Report beginning and ending of periods by the date of the first and last posting occurring in that period.

--exchange "*COMMODITY* [, *COMMODITY*, ...]" (**-X**)
Display values in terms of the given *COMMODITY*. If multiple commodities are given, values in a listed commodity will remain as-is, and others will be displayed in the first listed commodity they can be converted to.

--file *FILE* (**-f**)
Read journal data from *FILE*.

--first *INT*
Print the first *INT* entries. Opposite of **--last** *INT*. Alias for **--head**.

--flat Force the full names of accounts to be used in the balance report. The balance report will not use an indented tree.

--force-color
Output TTY color codes even if the TTY doesn't support them. Useful for TTYs that don't advertise their capabilities correctly.

--force-pager
Force **ledger** to paginate its output.

--forecast-while *EXPR*
Continue forecasting while *VEXPR* is true. Alias for **--forecast**.

--forecast-years *INT*
Forecast at most *INT* years into the future.

--format *FMT* (**-F**)
Use the given format string *FMT* to print output.

--gain (**-G**)
Report net gain or loss for commodities that have a price history.

--generated
Include auto-generated postings (such as those from automated transactions) in the report, in cases where you normally wouldn't want them.

--group-by *EXPR*
Group transaction together in the **register** report. *EXPR* can be anything, although most common would be *payee* or *commodity*. The **tag**() function is also useful here.

--group-title-format *FMT*
Set the format for the headers that separate reports section of a grouped report. Only has effect with a **--group-by** *EXPR* register report.

--hashes *ALGO*
Record the hash of each transaction in a *Hash* metadata value, according to the hashing algorithm given by the *ALGO* argument. Supported algorithms are *sha512* or *sha512_half* where both use SHA512, but the latter only stores the first half of the hash value.

If a *Hash* metadata value is explicitly provided and does not match what would have been generated, an error is reported. Hashes depend on previous entries, such that setting a single hash value is sufficient to guarantee the shape of the entire history leading up to that entry.

--head *INT*
Print the first *INT* entries. Opposite of **--tail** *INT*. Alias for **--first**

--help Print this man page.

--immediate
Evaluate calculations immediately rather than lazily.

--import *FILE*
Import *FILE* as Python module.

--init-file *FILE* (**-i**)
Read *FILE* before any other **ledger** file. This file may not contain any postings, but it may contain option settings. To specify options in the init file, use the same syntax as the command-line, but put each option on its own line.

--inject *STR*
Use *STR* amounts in calculations. In case you know what amount a transaction should be, but the actual transaction has the wrong value you can use metadata *STR* to specify the expected amount.

--input-date-format *DATEFMT*
Specify the input date format for journal entries.

--invert
Change the sign of all reported values.

--last *INT*.
Report only the last *INT* entries. Opposite of **--first** *INT*. Only useful on a register report. Alias for **--tail**.

--leeway *INT* (**-Z**)
Alias for **--price-exp**.

--limit *EXPR* (**-l**)
Limit postings in calculations.

--lisp-date-format *FORMAT*
Specify the format for dates in lisp output. The default uses the Emacs Lisp time format. Other options are ‘seconds’ (Unix epoch seconds) or a strftime format string like ‘%Y-%m-%d’.

--lot-dates
Report the date on which each commodity in a balance report was purchased.

--lot-notes
Report the tag attached to each commodity in a balance report.

--lot-prices

Report the price at which each commodity in a balance report was purchased.

--lots Report the date and price at which each commodity was purchased in a balance report.

--lots-actual
Preserve the uniqueness of commodities so they aren’t merged during reporting without printing the lot annotations.

--market (**-V**)
Use the latest market value for all commodities.

--master-account *STR*
Prepend all account names with *STR*

--meta *STR*
In the register report, prepend the transaction with the value of the given tag *STR*.

--meta-width *INT*
Specify the width of the Meta column used for the **--meta** *TAG* options.

--monthly (**-M**)
Shorthand for **--period** *monthly*.

--no-aliases
Aliases are completely ignored.

--no-color
Suppress any color TTY output.

--no-pager
Disables the pager on TTY output.

--no-revalued
Stop **ledger** from showing <Revalued> postings.

--no-rounding
Don’t output "<Adjustment>" postings. Note that this will cause the running total to often not add up! Its main use is for **--amount-data** (**-j**) and **--total-data** (**-J**) reports.

--no-titles

Suppress the output of group titles.

--no-total

Suppress printing the final total line in a balance report.

--now *DATE*

Use *DATE* as the current date. This affects the output when using **--period**, **--begin**, **--end**, or **--current** to decide which dates lie in the past or future.

--only *EXPR*

This is a postings predicate that applies after certain transforms have been executed, such as periodic gathering.

--options

Display the options in effect for this **ledger** invocation, along with their values and the source of those values.

--output *FILE* (**-o**)

Redirect the output of **ledger** to *FILE*.

--pager *STR*

Use *STR* as the pager program.

--payee

Sets a value expression for formatting the payee. In the **register** report this prevents the second entry from having a date and payee for each transaction.

--payee-width *INT*

Set the number of columns dedicated to the payee in the register report to *INT*.

--pedantic

Accounts, tags or commodities not previously declared will cause errors.

--pending

Use only postings that are marked pending.

--percent (**-%**)

Calculate the percentage value of each account in a balance reports. Only works for account that have a single commodity.

--period *PERIOD* (**-p**)
Define a period expression that sets the time period during which transactions are to be accounted. For a **register** report only the transactions that satisfy the period expression with be displayed. For a balance report only those transactions will be accounted in the final balances.

--period-sort
Sort the posting within transactions using the given value expression.

--permissive
Quiet balance assertions.

--pivot *TAG*
Produce a balance pivot report "around" the given *TAG*.

--plot-amount-format *FMT*
Define the output format for an amount data plot.

--plot-total-format *FMT*
Define the output format for a total data plot.

--prepend-format *FMT*
Prepend *FMT* to every line of the output.

--prepend-width *INT*
Reserve *INT* spaces at the beginning of each line of the output.

--price (**-I**)
Use the price of the commodity purchase for performing calculations.

--price-db *FILE*

--price-exp *INT* (**-Z**)
Set the expected freshness of price quotes, in *INT* hours. That is, if the last known quote for any commodity is older than this value, and if **--download** is being used, then the Internet will be consulted again for a newer price. Otherwise, the old price is still considered to be fresh enough. Alias for **--leeway**.

--prices-format *FMT*
Set the format for the **prices** report.

--pricedb-format *FMT*
Set the format expected for the historical price file.

--primary-date
Show primary dates for all calculations. Alias for **--actual-dates**

--quantity (**-O**)
Report commodity totals (this is the default).

--quarterly
Shorthand for **--period** *quarterly*.

--raw In the **print** report, show transactions using the exact same syntax as specified by the user in their data file. Don't do any massaging or interpreting. Can be useful for minor cleanups, like just aligning amounts.

--real (**-R**)
Account using only real transactions ignoring virtual and automatic transactions.

--recursive-aliases
Causes **ledger** to try to expand aliases recursively, i.e. try to expand the result of an earlier expansion again, until no more expansions apply.

--register-format *FMT*
Define the output format for the **register** report.

--related (**-r**)
In a register report show the related account. This is the other *side* of the transaction.

--related-all
Show all postings in a transaction, similar to **--related** but show both sides of each transaction.

--revalued
Report discrepancy in values for manual reports by inserting <Revalued> postings. This is implied when using the **--exchange** (**-X**) or **--market** (**-V**) option.

--revalued-only
Show only <Revalued> postings.

--revalued-total

Display the sum of the revalued postings as the running total, which serves to show unrealized capital in a gain/losses report.

--rich-data
When generating a ledger transaction from a CSV file using the **convert** command, add CSV, Imported, and UUID meta-data.

--seed *INT*
Set the random seed to *INT* for the **generate** command. Used as part of development testing.

--script *FILE*
Execute a **ledger** script.

--sort *EXPR* (**-S**)
Sort the register report based on the value expression *EXPR*.

--sort-xacts
Sort the posting within transactions using the given value expression.

--start-of-week *STR*
Use *STR* as the particular day of the week to start when using the **--weekly** option. *STR* can be day names, their abbreviations like "Mon", or the weekday number starting at 0 for Sunday.

--sort-all
Sort all accounts and commodities.

--strict Accounts, tags or commodities not previously declared will cause warnings.

--subtotal (**-s**)
Report register as a single subtotal.

--tail *INT*
Report only the last *INT* entries. Only useful on a register report. Alias for **--last** *INT*

--time-colon
Display the value for commodities based on seconds as hours and minutes. Thus 8100s will be displayed as 2:15h instead of 2.25h.

--time-report
Add two columns to the **balance** report to show the earliest checkin and checkout times for

timelog entries.

--total *EXPR* (**-T**)
Define a value expression used to calculate the total in reports.

--total-data (**-J**)
Show only dates and totals to format the output for plots.

--total-width *INT*
Set the width of the total field in the register report.

--trace *INT*
Enable tracing. The *INT* specifies the level of trace desired.

--truncate *STR*
Indicates how truncation should happen when the contents of columns exceed their width. Valid arguments for *STR* are *leading*, *middle*, and *trailing*. The default is smarter than any of these three, as it considers sub-names within the account name (that style is called "abbreviate").

--unbudgeted
Show only un-budgeted postings.

--uncleared (**-U**)
Use only uncleared transactions in calculations and reports.

--unrealized
Show generated unrealized gain and loss accounts in the balance report.

--unrealized-gains
Allow the user to specify what account name should be used for unrealized gains. Defaults to **Equity:Unrealized Gains**. Often set in one’s init file to change the default.

--unrealized-losses
Allow the user to specify what account name should be used for unrealized losses. Defaults to **Equity:Unrealized Losses**. Often set in one’s init file to change the default.

--unround
Perform all calculations without rounding and display results to full precision.

--values

Show the values used by each tag when used in combination with the **tags** command.

--value-expr *EXPR*
Set a global value expression annotation.

--verbose
Print detailed information on the execution of **ledger**.

--verify
Enable additional assertions during run-time. This causes a significant slowdown. When combined with **--debug** *CODE* **ledger** will produce memory trace information.

--verify-memory
Verify that every constructed object is properly destructed. This is for debugging purposes only.

--version
Print version information and exit.

--weekly (**-W**)
Shorthand for **--period** *weekly*.

--wide (**-w**)
Assume 132 columns instead of the TTY width.

--yearly (**-Y**)
Shorthand for **--period** *yearly*.

PRE-COMMANDS

Pre-commands are useful when you aren't sure how a command or option will work. The difference between a pre-command and a regular command is that pre-commands ignore the journal data file completely, nor is the user's init file read.

args / **query**
Evaluate the given arguments and report how **ledger** interprets it against the following model transaction:

```
2004/05/27 Book Store
  ; This note applies to all postings. :SecondTag:
  Expenses:Books              20 BOOK @ $10
```

```
; Metadata: Some Value
; Typed:: $100 + $200
; :ExampleTag:
; Here follows a note describing the posting.
Liabilities:MasterCard        $-200.00
```

eval Evaluate the given value expression against the model transaction.

format Print details of how **ledger** uses the given formatting description and apply it against a model transaction.

parse / **expr**
Print details of how **ledger** uses the given value expression description and apply it against a model transaction.

generate
Randomly generates syntactically valid **ledger** data from a seed. Used by the GenerateTests harness for development testing.

period Evaluate the given period and report how **ledger** interprets it.

template
Shows the insertion template that the **xact** command generates. This is a debugging command.

QUERIES

The syntax for reporting queries can get somewhat complex. It is a series of query terms with an implicit OR operator between them. The following terms are accepted:

regex A bare string is taken as a regular expression (PCRE) matching the full account name. Thus, to report the current balance for all assets and liabilities, you would use:

```
ledger bal asset liab
```

payee *regex* (**@***regex*)
Query on the payee, rather than the account.

tag *regex* (**%***regex*)
Query on tags.

note *regex* (**=***regex*)

Query on anything found in an item's note.

code *regex* (**#***regex*)
Query on the xact's optional code (which can be any string the user wishes).

term **and** *term* Query terms are joined by an implicit OR operator. You can change this to AND by using the **and** keyword. For example, to show food expenditures occurring at Shakee's Pizza, you could say:

ledger reg food and @Shakee

term **or** *term* When you wish to be more explicit, use the OR operator.

show

not *term* Reverse the logical meaning of the following term. This can be used with parentheses to great effect:

ledger reg food and @Shakee and not dining

(*term*) If you wish to mix OR and AND operators, it is often helpful to surround logical units with parentheses. **NOTE**: Because of the way some shells interpret parentheses, you should always escape them:

ledger bal \(assets or liab \) and not food

EXPRESSIONS

abs(*value*) Return the absolute value of the given *value*.

account Return the posting's account.

account_base Return the base account, i.e. everything after the last account delimiter ':'.

actual Return true if the transaction is real, i.e not an automated or virtual transaction, false otherwise.

amount Return the amount of the posting.

amount_expr Return the calculated amount of the posting according to the **--amount** option.

ansify_if(*value*, *color*, *bool*)
Render the given *value* as a string, applying the proper ANSI escape codes to display it in the given *color* if *bool* is true. It typically checks the value of the option **--color**, for example:
ansify_if(amount, blue, options.color)

beg_line Line number where entry for posting begins.

beg_pos Character position where entry for posting begins.

ceiling(*value*) Return the next integer of *value* toward +infinity.

cleared Return true if the posting was cleared, false otherwise.

code Return the transaction code, the string between the parenthesis after the date.

commodity(*value*)
Return the commodity of *value* or the posting amount when *value* was not specified.

date Return the date of the posting.

end_line Line number where entry for posting ends.

end_pos Character position where entry for posting ends.

floor(*value*) Return the next integer of *value* toward -infinity.

filename The name of the **ledger** data file from whence the posting came.

format(*string*) Evaluate *string* as format just like the **--format** option.

format_date(*date*, *format*)
Return the *date* as a string using *format*. Refer to strftime(3) for format string details.

format_datetime(*datetime*, *format*)
Return the *datetime* as a string using *format*. Refer to strftime(3) for format string details.

get_at(*seq*, *index*)
Return value at *index* from *seq*. Used internally to construct different reports.

has_meta(*tag*) Return true if the posting has metadata named *tag*, false otherwise.

has_tag(*tag*) Return true if the posting has metadata named *tag*, false otherwise.

is_seq(*value*) Return true if *value* is a sequence. Used internally.

join(*value*) Replace all newlines in *value* with \n.

justify(*value, first_width, latter_width, right_justify, colorize*)
Right or left justify the string representing *value*. The width of the field in the first line is given by *first_width*. For subsequent lines the width is given by *latter_width*. If *latter_width* is -1, *first_width* is used for all lines. If *right_justify* is true then the field is right justified within the width of the field. If it is false, then the field is left justified and padded to the full width of the field. If *colorize* is true, then ledger will honor color settings.

market(*value*, *datetime*)
Return the price of *value* at *datetime*. Note that *datetime* must be surrounded by brackets in order to be parsed correctly, e.g. [2012/03/23].

meta(*name*) Return the value of metadata named *name*.

note Return the note for the posting.

now Return the current datetime.

options A variable that allows access to the values of the given command-line options using the long option names, e.g. to see whether **--daily** (**-D**) was given use **option.daily**.

payee Return the payee of the posting.

percent(*value_a, value_b*)
Return the percentage of *value_a* in relation to *value_b* (used as 100%).

pending Return true if the posting is marked as pending, false otherwise.

percent(*value_a, value_b*)
Return the percentage of *value_a* in relation to *value_b*.

print(*value*) Print *value* to stdout. Used internally for debugging.

quantity(*value*) Return the quantity of *value* for values that have a per-unit cost.

quoted(*expression*)
Surround *expression* with double quotes.

quoted_rfc(*expression*)
Surround *expression* with double quotes, compliant with rfc 4180.

real Return true if the transaction is real, i.e not an automated or virtual transaction, false otherwise.

roundto(*value, n*)
Return *value* rounded to *n* digits. Does not affect formatting.

should_bold Return true if expression given to **--bold-if** evaluates to true. Internal use only!

scrub(*value*) Clean *value* using various transformations such as round, stripping value annotations, and more.

strip(*value*) Strip value annotation from *value*.

tag(*name*) Return the value of tag named *name*.

to_amount(*value*)
Convert *value* to an amount. Internal use only!

to_balance(*value*)
Convert *value* to a balance. Internal use only!

to_boolean(*value*)
Convert *value* to a boolean. Internal use only!

to_date(*value*) Convert *value* to a date. Internal use only!

to_datetime(*value*)
Convert *value* to a datetime. Internal use only!

to_int(*value*) Return the integer value for *value*.

to_mask(*value*)

Convert *value* to a mask. Internal use only!

to_sequence(*value*)
Convert *value* to a sequence. Internal use only!

to_string(*value*)
Convert *value* to a character string.

today Return today's date.

total Return the total of the posting.

total_expr Return the calculated total of the posting according to the **--total** option.

trim(*value*) Trim leading and trailing whitespace from *value*.

truncated(*string*, *total_len*, *account_len*)
Truncate *string* to *total_len* ensuring that each account is at least *account_len* long.

virtual Return true if the transaction is virtual, e.g automated, false otherwise.

averaged_lots()
Return lots with averaged prices.

clear_commodity()
Clear commodity pricing information.

commodity_price(*commodity*)
Return the price of the given commodity.

display_amount(*amount*)
Format amount for display.

display_total(*total*)
Format total for display.

lot_date(*posting*)
Return the lot date of a posting.

lot_price(*posting*)

Return the lot price of a posting.

lot_tag(*posting*)
Return the lot tag of a posting.

nail_down(*amount*)
Fix the precision of an amount.

round(*amount*) Round an amount to standard precision.

rounded(*amount*)
Return rounded version of amount.

set_commodity_price(*commodity*, *price*)
Set the price for a commodity.

top_amount(*amount*)
Return the top-level amount.

unround(*amount*)
Remove rounding from an amount.

unrounded(*amount*)
Return unrounded version of amount.

DEBUG COMMANDS

In addition to the regular reporting commands, **ledger** also accepts several debug commands:

args [*report-query*]
Display complete analysis of how **ledger** interpreted the given *report-query*. Useful if you want to understand how report queries are translated into value expressions.

eval [*value-expression*]
Evaluate the given *value-expression* and prints the result. For more on value expressions, see the section *EXPRESSIONS*.

format [*format-string*]
Display an analysis of how *format-string* was parsed, and what it would look like applied to a sample transaction. For more on format strings, see the section *FORMATS*.

generate
Generate 50 randomly composed yet valid **ledger** transactions.

parse [*value-expression*]
Parse the given *value-expression* and display an analysis of the expression tree and its evaluated value. For more on value expressions, see the section *EXPRESSIONS*.

python [*file*]
Invoke a Python interpreter to read the given *file*. What is special about this is that the **ledger** module is builtin, not read from disk, so it doesn't require **ledger** to be installed anywhere, or the shared library variants to be built.

reload Reload all data files for the current session immediately. Can only be used in the REPL.

template [*draft-template*]
Display information about how *draft-template* was parsed. See the section on *DRAFTS*.

ENVIRONMENT

Every option to **ledger** may be set using an environment variable if the option has a long name. For example setting the environment variable LEDGER_DATE_FORMAT="%d.%m.%Y" will have the same effect as specifying **--date-format** *'%d.%m.%Y'* on the command-line. Options on the command-line always take precedence over environment variable settings, however.

FILES

$XDG_CONFIG_HOME/ledger/ledgerrc

~/.config/ledger/ledgerrc

~/.ledgerrc
Your personal **ledger** initializations.

SEE ALSO

beancount(1), hledger(1)

The full documentation for **ledger** is maintained as a Texinfo manual. If the **info** program is installed on your system, the command
info ledger3
should give you access to the complete manual.

AUTHORS

John Wiegley <johnw@newartisans.com>