Running Beancount & Generating Reports

Martin Blais, July-Sep 2014

http://furius.ca/beancount/doc/tools

IMPORTANT: This document applies to tools from the v2 branch which have been deprecated. Many of these tools have been deleted for v3 and beyond.

Introduction

This document describes the tools you use to process Beancount input files, and many of the reports available from it. The syntax of the language is described in the Beancount Language Syntax document. This manual only covers the technical details for using Beancount from the command-line.

Tools

bean-check

bean-check is the program you use to verify that your input syntax and transactions work correctly. All it does is load your input file and run the various plugins you configured in it, plus some extra validation checks. It report errors (if any), and then exits. You run it on your input file, like this:

```
bean-check /path/to/my/file.beancount
```

If there are no errors, there should be no output, it should exit quietly. If there were errors, they will be printed to stderr with the filename, line number and error description (in a format that is understood by Emacs, so you can just use `next-error` and `previous-error` to navigate to the file if you want):

```
/home/user/myledger.beancount:44381:   Transaction does not balance: 34.46 USD

   2014-07-12 * "Black Iron Burger" ""
     Expenses:Food:Restaurant               17.23 USD
     Assets:Cash                            17.23 USD
```

You should always fix all the errors before producing reports.

bean-report

This is the main tool used to extract specialized reports to the console in text or one of the various other formats. You invoke it like this:

```
bean-report /path/to/my/file.beancount <report-name>
```

For example:

```
bean-report /path/to/my/file.beancount balances
```

There are many reports available. See the section on reports for a description of the main ones. If you want to produce the full list of reports, ask it for help:

```
bean-report --help-reports
```

Report names can sometimes accept arguments. At the moment the arguments are specified as part of the report name itself, often separated by a colon (:), like this:

```
bean-report /path/to/my/file.beancount balances:Vanguard
```

There are a few special reports you should know about:

- `check`, or `validate`: This is the same as running the **bean-check** command.
- `print`: This simply prints out the entries that Beancount has parsed, in Beancount syntax. This can be used to confirm that Beancount has read and interpreted your input data

correctly (if you're debugging something difficult).

The other reports are what you'd expect: they print out various tables of aggregations of amounts. The reports you can generate are described in a dedicated section below.

> ***PLEASE NOTE!*** At the moment of release, the list of reports available from the web page will differ from the list available from the console. In a future release, I will consolidate those two lists and all the reports that are available from the web pages will also be available from the console, and in many different formats. Stay tuned.

bean-query

Beancount's parsed list of transactions and postings is like an in-memory database. **bean-query** is a command-line tool that acts like a client to that in-memory database in which you can type queries in a variant of SQL. You invoke it like this:

```
bean-query /path/to/my/file.beancount
Input file: "/path/to/my/file.beancount"
Ready with 14212 directives (21284 postings in 8879 transactions).
beancount> _
```

More details are available in its own document.

bean-web

bean-web serves all the reports on a web server that runs on your computer. You run it like this:

```
bean-web /path/to/my/file.beancount
```

It will serve all pages on port 8080 on your machine. Navigate to http://localhost:8080 with a web browser. You should be able to click your way through all the reports easily.

The web interface provides a set of global pages and a set of report pages for each "view."

Global Pages

The top-level table of contents page provides links to all the global pages at the top:

- The table of contents (the page you're looking at)
- A list of the errors that occurred in your ledger file
- A view of the source code of your Ledger file (this is used by various other links when referring to a location in your input file.)

The table of contents provides a convenient list of links to all the common views, such as

- "view by year",
- "view by tag", and of course,
- "view all transactions."

There are a few more.

View Reports Pages

When you click on a view report page, you enter a set of pages for the subset of transactions for that

view. Various reports about those transactions are available from here:

- Opening balances (a balance sheet at the beginning of the view)
- Balance sheet (a balance sheet at the end of the view)
- Income statement (for the period of the view)
- Various journals for each account (just click on an account name)
- Various reports of holdings at the end of the view
- Lists of documents and prices included in the view's entries
- Some statistics about the view data

… and much more. There should be an index of all the available view reports.

bean-bake

bean-bake runs a **bean-web** instance and bakes all the pages to a directory:

```
bean-bake /path/to/my/file.beancount myfinances
```

It also support baking directly to an archive file:

```
bean-bake /path/to/my/file.beancount myfinances.zip
```

Various compression methods are supported, e.g. .tar.gz.

This is useful to share the web interface with your accountant or other people, who usually don't have the ability to run Beancount. The page links have all been converted to relative links, and they should be able to extract the archive to a directory and browse all the reports the same way you do with **bean-web**.

bean-doctor

This is a debugging tool used to perform various diagnostics and run debugging commands, and to help provide information for reporting bugs. For example, it can do the following:

- List the Beancount dependencies that are installed on your machine, and the ones that are missing. It can tell you what you're missing that you should be installing.
- Print a dump of the parser's lexer tokens for your input file. If you report a parsing bug, it can be useful to look at the lexer output (if you know what you're doing).
- It can run your input file through a parsing round-trip, that is, print out the file and re-read it again and compare it. This is a useful parser and printer test.
- It can check that a directory hierarchy corresponds to a Beancount input file's chart-of-accounts, and reporting directories that do not comply. This is useful in case you decide to change some account names and are maintaining a corresponding archive of documents which needs to be adjusted accordingly.

Context

It can list the context upon which a transaction is applied, i.e., the inventory balances of each account the transaction postings modify, before and after it is applied. Use it like this:

```
$ bean-doctor context /home/blais/accounting/blais.beancount 28514
/home/blais/accounting/blais.beancount:28513:

;   Assets:US:ETrade:BND             100.00 BND {81.968730000000 USD}
```

```
;   Assets:US:ETrade:Cash           8143.97 USD

2014-07-25 * "(TRD) BOT +50 BND @82.10" ^273755872
 Assets:US:ETrade:BND                 50.00 BND {82.10 USD}
 Assets:US:ETrade:Cash             -4105.00 USD

;   Assets:US:ETrade:BND                   100.00 BND {81.968730000000 USD}

; ! Assets:US:ETrade:BND               50.00 BND {82.10 USD}
; ! Assets:US:ETrade:Cash            4038.97 USD
```

There is a corresponding Emacs binding (C-c x) to invoke this around the cursor.

bean-format

This pure text processing tool will reformat Beancount input to right-align all the numbers at the same, minimal column. It left-aligns all the currencies. It only modifies whitespace. This tool accepts a filename as arguments and outputs the aligned file on stdout (similar to UNIX `cat`).

bean-example

This program generates an example Beancount input file. See the Tutorial for more details about the contents of this example file.

Filtering Transactions

In order to produce different views of your financial transactions, we select a subset of full list of parsed transactions, for example, “all the transactions that occurred in year 2013”, and then use that to produce the various available reports that Beancount provides.

At the moment, only a preset list of filters are available as “views” from the web interface. These views include:

- All transactions
- Transactions that occur in a particular year
- Transactions with a particular tag
- Transactions with a particular payee
- Transactions that involve at least one account with a particular name component

At the moment, in order to access reports from these subsets of transactions, you need to use the **bean-web** web interface, and click on the related keyword in the root global page, which enters you into a set of reports for that view.

Reports

The whole point of entering your transactions in a single input file in the first place is that it allows you to sum, filter, aggregate and arrange various subsets of your data into well-known reports. There are three distinct ways to produce reports from Beancount: by using **bean-web** and browsing to a view and then to a specific report (this is the easy way), by using **bean-report** and providing the name of a desired report (and possibly some report-specific arguments), and by using **bean-query** and requesting data by specifying an SQL statement.

Reports can sometimes be rendered in different file formats. Each report type will support being

rendered in a list of common ones, such as console text, HTML and CSV. Some reports render in Beancount syntax itself, and we simply call this format name "beancount."

There are many types of reports available, and there will be many more in the future, as many of the features on the roadmap involve new types of output. This section provides an overview of the most common ones. Use bean-report to inquire about the full list supported by your installed version:

```
bean-report --help-reports
```

> ***PLEASE NOTE!*** At the moment, the sets of reports that are available from the web interface and from the console are different, though there is some overlap. In a subsequent version, the list of reports will be reconciled and all reports will be made available via both the web interface and the console, in a variety of data formats (text, CSV, HTML and maybe others.) For now, we will document these in the sections below.

Balance Reports

All the balance reports are similar in that they produce tables of some set of accounts and their associated balances: [output]

```
|-- Assets
|   `-- US
|       |-- BofA
|       |   `-- Checking                      596.05 USD
|       |-- ETrade
|       |   |-- Cash                        5,120.50 USD
|       |   |-- GLD                            70.00 GLD
|       |   |-- ITOT                           17.00 ITOT
|       |   |-- VEA                            36.00 VEA
|       |   `-- VHT                           294.00 VHT
|       |-- Federal
|       |   `-- PreTax401k
|       |-- Hoogle
|       |   `-- Vacation                      337.26 VACHR
|       `-- Vanguard
...
```

Balances for commodities held "at cost" are rendered at their book value. (Unrealized gains, if any, are inserted as separate transactions by an optional plugin and the result amounts get mixed in with the cost basis if rendered in the same account.)

If an account's balance contains many different types of currencies (commodities not held "at cost", such as dollars, euros, yen), each gets printed on its own line. This can render the balance column a bit too busy and messes with the vertical regularity of account names. This is to some extent an unavoidable compromise, but in practice, there are only a small number of commodities that form the large majority of a user's ledger: the home country's currency units. To this extent, amounts in common currencies can be broken out into their own column using the "`operating_currency`" option:

```
option "operating_currency" "USD"
option "operating_currency" "CAD"
```

You may use this option multiple times if you have many of them (this is my case, for instance, because I am an expat and hold assets in both my host and home countries). Declaring operating

currencies also hold the advantage that the name of the currency need not be rendered, and it can thus more easily be imported into a spreadsheet.

Finally, some accounts are deemed "active" if they have not been closed. A closed account with no transactions in the filtered set of transactions will not be rendered.

Trial Balance (`balances`)

A trial balance report simply produces a table of final balances of all the active accounts, with all the accounts rendered vertically.

The sum total of all balances is reported at the bottom. Unlike a balance sheet, it may not always balance to zero because of currency conversions. (This is the equivalent of Ledger's `bal` report.)

The equivalent bean-query command is:

```
SELECT account, sum(position)
GROUP BY account
ORDER BY account;
```

Balance Sheet (`balsheet`)

A balance sheet is a snapshot of the balances of the Assets, Liabilities and Equity accounts at a particular point in time. In order to build such a report, we have to move balances from the other accounts to it:

1. compute the balances of the Income and Expenses account at that point in time,
2. insert transactions that will zero out balances from these accounts by transferring them to an equity account (`Equity:Earnings:Current`),
3. render a tree of the balances of the Assets accounts on the left side,
4. render a tree of the Liabilities accounts on the right side,
5. render a tree of the Equity accounts below the Liabilities accounts.

See the introduction document for an example.

Note that the Equity accounts include the amounts reported from the Income and Expenses accounts, also often called "Net Income."

Also, in practice, we make *two* transfers because we're typically looking at a reporting period, and we want to differentiate between Net Income amounts transferred before the beginning of the period (`Equity:Earnings:Previous`) and during the period itself (`Equity:Earnings:Current`). And similar pair of transfers is carried out in order to handle currency conversions (this is a bit of a hairy topic, but one with a great solution; refer to the dedicated document if you want all the details).

The equivalent bean-query command is:

```
SELECT account, sum(position)
FROM CLOSE ON 2016-01-01
GROUP BY account ORDER BY account;
```

Opening Balances (`openbal`)

The opening balances report is simply a balance sheet drawn at the beginning of the reporting period. This report only makes sense for a list of filtered entries that represents a period of time, such as "year 2014." The balance sheet is generated using only the summarization entries that were synthesized when the transactions were filtered (see the double-entry method document).

Income Statement (`income`)

An income statement lists the final balances of the Income and Expenses accounts. It represents a summary of the transient activity within these accounts. If the balance sheet is the snapshot at a particular point in time, this is the difference between the beginning and the end of a period (in our case: of a filtered set of transactions). The balances of the active Income accounts are rendered on the left, and those of the active Expenses accounts on the right. See the introduction document for an example.

The difference between the total of Income and Expenses balances is the Net Income.

Note that the initial balances of the Income and Expenses accounts should have been zero'ed out by summarization transactions that occur at the beginning of the period, because we're only interested in the changes in these accounts.

Journal Reports

The reports in this section render lists of transactions and other directives in a linear fashion.

Journal (`journal`)

This report is the equivalent of an "account statement" from an institution, a list of transactions with at least one posting in that account. This is the equivalent of Ledger's register report (`reg`).

You generate a journal report like this:

```
bean-report myfile.beancount journal …
```

By default, this renders a journal of *all* the transactions, which is unlikely to be what you want. Select a particular account to render like this:

```
bean-report myfile.beancount journal -a Expenses:Restaurant
```

At the moment, the "`-a`" option accepts only a complete account name, or the name of one of the parent accounts. Eventually we will extend it to handle expressions.

Rendering at Cost

The numbers column on the right displays the changes from the postings of the selected account. Notice that only the balances for the postings affecting the given account are rendered.

The change column renders the changes in the units affected. For example, if this posting is selected:

```
Assets:Investments:Apple      2 AAPL {402.00 USD}
```

The value reported for the change will be "`2 AAPL`". If you would like to render the values at cost, use the "`--at-cost`" or "`-c`" option, which will in this case render "`804.00 USD`" instead.

There is no "market value" option. Unrealized gains are automatically inserted at the end of the history by the "`beancount.plugins.unrealized`" plugin. See options for that plugin to insert its unrealized gains.

Note that if the sum of the selected postings is zero, no amount is rendered in the change column.

Adding a Balance Column

If you want to add a column that sums up the running balance for the reported changes, use the "`--render-balance`" or "`-b`" option. This does not always make sense to report, so it is up to you to decide whether you want a running balance.

Character Width

By default, the report will be as wide as your terminal allows. Restrict the width to a set number of characters with the "-w" option.

Precision

The number of fractional digits for the number rendering can be specified via "`--precision`" or "`-k`".

Compact, Normal or Verbose

In its normal operation, Beancount renders an empty line between transactions. This helps delineate transactions where there are multiple currencies affected, as they render on separate lines. If you want a more compact rendering, use the "`--compact`" or "`-x`" option.

On the other hand, if you want to render the affected postings under the transaction line, use the "`--verbose`" or "`-X`" option.

Summary

Here is a summary of its arguments:

```
optional arguments:
  -h, --help            show this help message and exit
  -a ACCOUNT, --account ACCOUNT
                        Account to render
  -w WIDTH, --width WIDTH
                        The number of characters wide to render the report to
  -k PRECISION, --precision PRECISION
                        The number of digits to render after the period
  -b, --render-balance, --balance
                        If true, render a running balance
  -c, --at-cost, --cost
                        If true, render values at cost
  -x, --compact         Rendering compactly
  -X, --verbose         Rendering verbosely
```

Equivalent SQL Query

The equivalent bean-query command is:

```
SELECT date, flag, description, account, cost(position), cost(balance);
```

Conversions (`conversions`)

This report lists the total of currency conversions that result from the selected transactions. (Most people won't need this.)

Documents (`documents`)

This report produces an HTML list of all the external documents found in the ledger, either from explicit directives or from plugins that automatically find the documents and add them to the stream of transactions.

Holdings Reports

These reports produces aggregations for assets held at cost.

Holdings & Aggregations (`holdings*`)

This report produces a detailed list of all holdings found in the ledger. You can produce aggregates by commodity and accounts using the “-g” option.

Net Worth (`networth`)

This report produces a short summary of the net worth (equity) of the ledger, in each of the operating currencies.

Other Report Types

Cash

This report renders balances in commodities not held at cost, in other words, cash:

```
bean-report example.beancount cash -c USD

Account                        Units  Currency  Cost Currency  Average Cost  Price  Book Value  Market Value
-------------------------  ---------  --------  -------------  ------------  -----  ----------  ------------
Assets:US:BofA:Checking        596.05       USD            USD                          596.05        596.05
Assets:US:ETrade:Cash        5,120.50       USD            USD                        5,120.50      5,120.50
Assets:US:Hoogle:Vacation      337.26     VACHR
Assets:US:Vanguard:Cash         -0.02       USD            USD                           -0.02         -0.02
Liabilities:US:Chase:Slate  -2,891.85       USD            USD                       -2,891.85     -2,891.85
-------------------------  ---------  --------  -------------  ------------  -----  ----------  ------------
```

The report allows you to convert all currencies to a common currency (in the example above, "convert everything to USD"). There's also an option to report only on the operating currencies. I use this to get an overview of all uninvested cash.

Prices (`prices`)

This report renders a list of price points for a base currency in terms of a quote currency. The list is sorted by date. You can output this table in beancount format as well. This is convenient to save a price database to a file, that can then be combined and loaded into another input file.

Statistics (`stats`)

This report simply provides various statistics on the parsed entries.

Update Activity (`activity`)

This table renders for each account the date of the last entry.