diamond-accounting Documentation

Release 0.1.1rc1

Ian Dennis Miller

Oct 27, 2018

Contents

Diamond-Accounting helps you manage your books.

CHAPTER 1

Overview

1.1 Installation

Install diamond-accounting with Python pip.

```
pip install diamond-accounting
```

1.2 Usage

Create an accounting file structure in the current folder.

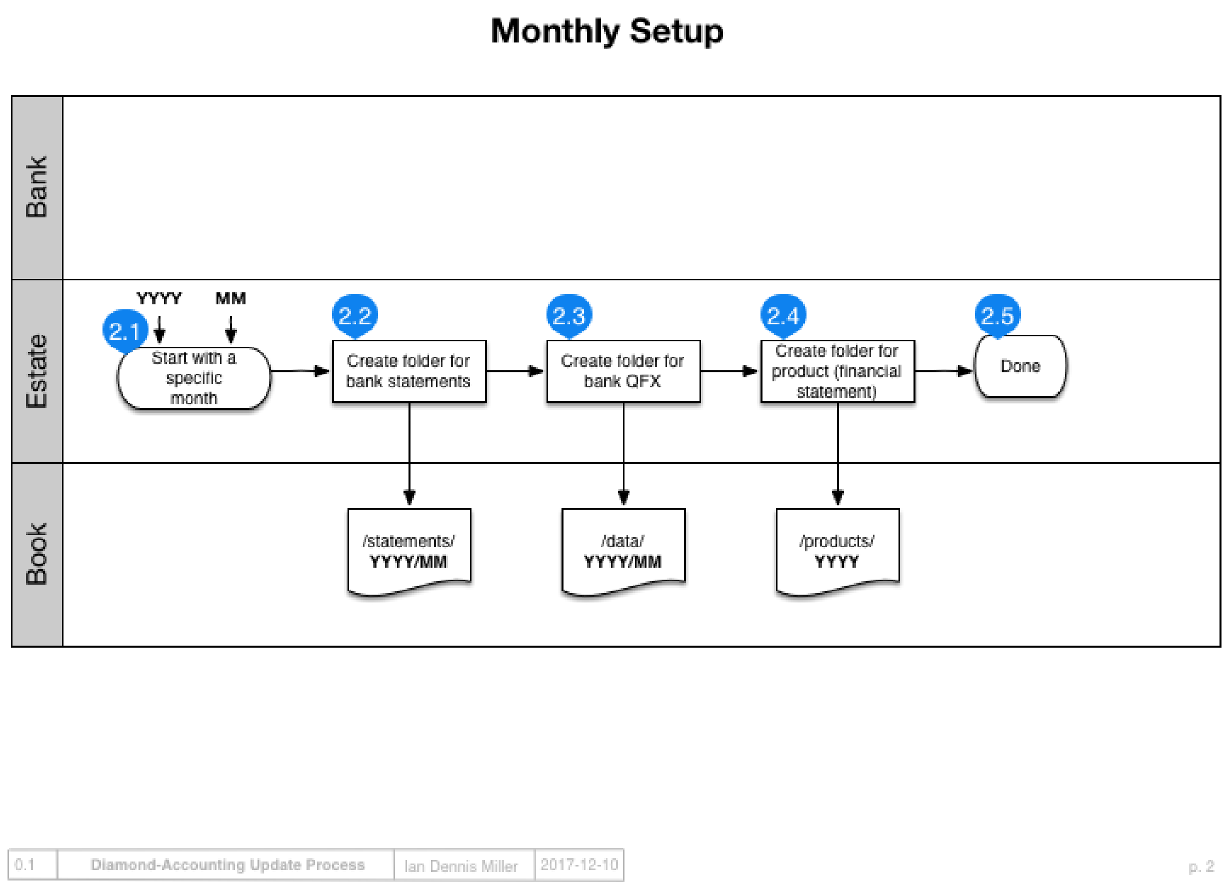
```
diamond-accounting scaffold
```

1.3 Documentation

http://diamond-accounting.readthedocs.io

1.3.1 Bookkeeping

Monthly Setup



Setup

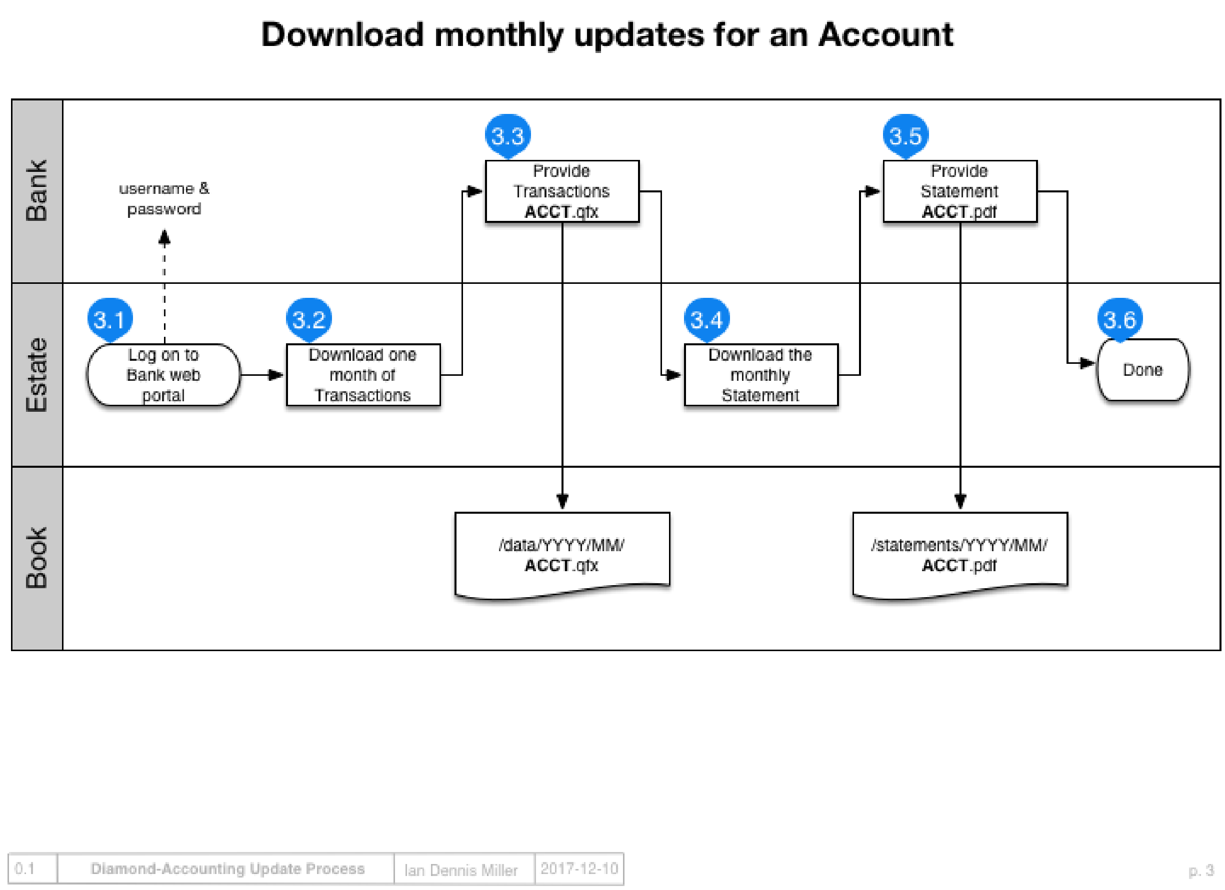
Setup the ledger project environment. This ensures the right software is open, the right files are open, and things are going to be ready for work.

1. Open password manager software.
2. Enter the ledger project with *project-workon ledger*
3. Load *ledgers/inbox.ledger* and *ledgers/budget.ledger* in editor.
4. Load *ledgers/YYYY/transfers.ledger* in editor for transfers between accounts.
5. Enter SublimeText 2-column mode with *cmd-alt-2*. Put the inbox on the right and everything else on the left.
6. Open the interactive register with *make ui* so that all accounts can be viewed and historical transactions can be explored.

Next steps

Next read about *Download Transactions*.

Download Transactions



The goal of downloading transactions is to download a *.QFX* file for each account containing one month of that account's transactions. These files will eventually be imported into the ledger with the sync process.

Throughout the download process, accounts are referred to by the last 4 or 5 digits of their account number. When an account is downloaded, the file will be renamed using the 4 or 5 digit number.

Unique portfolios

Each bank or financial institution has a unique process for downloading transactions. To make this process as repeatable as possible, it is recommended to use account management software to store web URLs and authentication information for logging into your various bank web portals. Since each accounting portfolio is different, downloading needs to be described in a separate document that will be different for each project. NB: the Readme.md file in a project scaffold has a starting place for creating this download documentation.

Moving downloads to ledger project

Once the *.QFX* files have been downloaded and renamed, they must be moved to the ledger project so they can be archived. Create a folder called *data/YYYY/MM* in the project, where **YYYY** is the year and **MM** is the month. Move all the downloaded *.QFX* files into this new folder and check it into the repository.

Download Statements

Each financial institution produces periodic statements that memorialize your account balances. These statements need to be downloaded or scanned so they can be compared to the ledger balance, later. The most convenient time to download these statements is when downloading transaction data. As with transaction data, the process for downloading statements is unique to each bank.

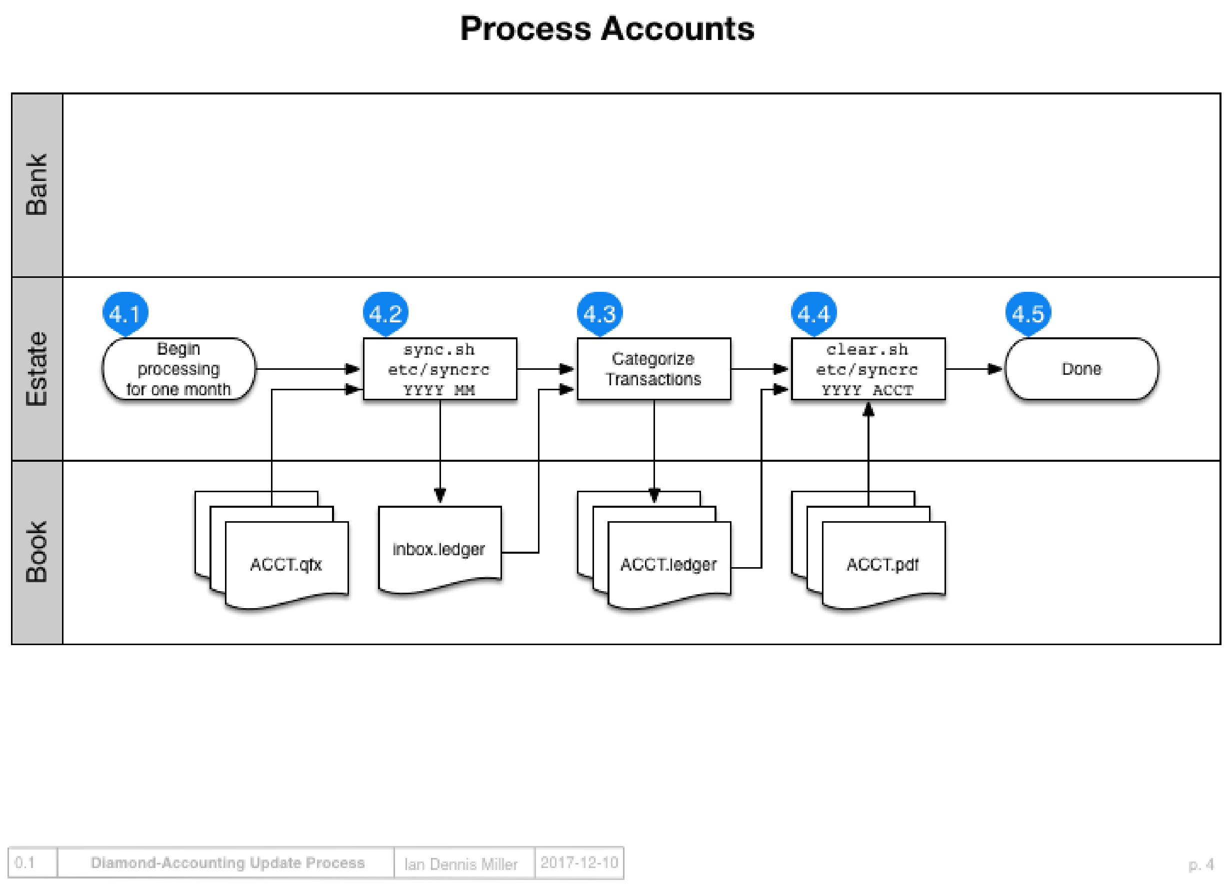
Downloading is complete

Once the *.QFX* and *.PDF* files are in the repository, the ledger project contains an archive of the bank's record of your account transactions. These archived *.QFX* and *.PDF* files are unlikely to change and, since they are computer-readable, they are a convenient source of raw transaction data for updating our ledgers.

Next steps

Next read about *Sync transactions*.

Sync transactions



Sync transactions from the downloaded files in `/data/` to Inbox. This corresponds to **step 4.2** in the diagram.

After downloading, transactions need to be synced to *ledgers/inbox.ledger* according to the configuration specified by [etc/syncrc](../etc/syncrc). The sync process expects to find *.QFX* files for import in within the ledger project itself, inside the *data/* folder. To perform the sync, use *sync.sh*, substituting the year and month as appropriate:

```
sync.sh etc/syncrc YYYY MM
```

As accounts are synced to the inbox, *sync.sh* adds a comment to the file to provide notice that a new account is being synced. Watch for these comments during import so you know when to switch ledger files. The *ofxid* that appears as a comment in each transaction must remain intact for re-running *sync.sh* to be possible.

The configuration file *syncrc*

The configuration file describes all of the accounts that should be imported by *sync.sh*. Each row in the configuration is one account and it has three fields:

- financial ID
- 4 or 5-digit account ID
- account name to import into ledger as.

For example:

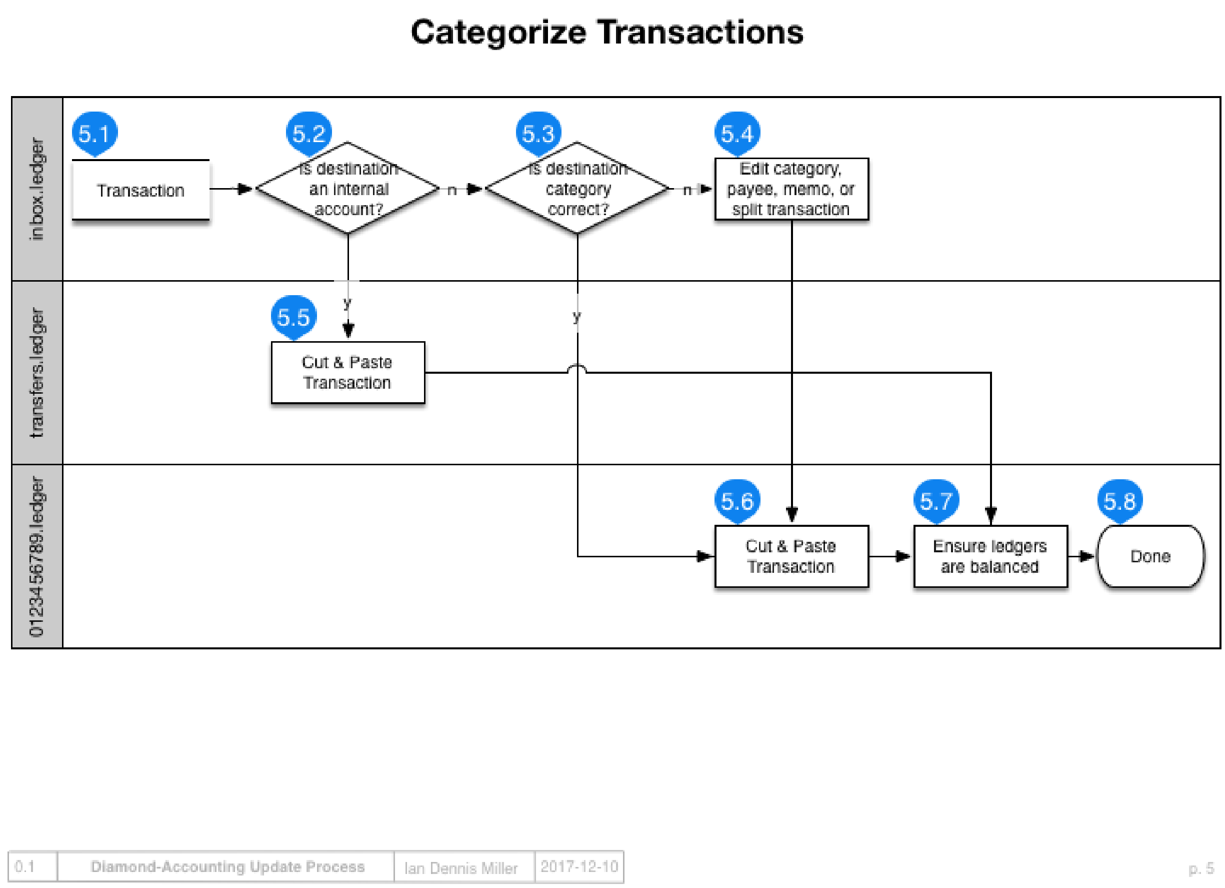
```
1 1234 Assets:BankOne:Checking
1 2345 Assets:BankOne:Savings
2 3456 Assets:BankTwo:Savings
```

The financial ID is a number that chosen by you, but it should be unique and consistent within your ledgers. The account ID and account name should match your portfolio.

Next steps

Next read about *Categorizing*.

Categorizing



Process Inbox

Once transactions have been synced to *inbox.ledger*, it is time to process the inbox. The user imports transactions by cut-and-pasting them from the *inbox* into the destination ledger. The destination ledger is opened in the text editor depending on the account is indicated by the *sync.sh* comments. When the inbox is empty because all the transactions were pasted, this process is done.

The primary goals for processing the inbox are: 1) ensure the categories of the transactions have been set correctly; and 2) migrate those transactions into the proper ledger. The category can usually be determined from the memo and the date, which should enable you to remember the transaction or find a receipt that can establish the category. There is a separate review step called "clearing" in which transactions are more closely examined to figure out if they are accurate or not.

Category Curation

It is important to consistently label and use your accounts. Use the account browser to quickly browse the available account names so you can keep your expense categories clean.

```
make ui
```

In case you forget the name of a category, it is simple to see some example expenses for a category with the user interface.

Rules for processing

There are a few things to watch for while processing.

- If the transaction is a transfer between your own ledger accounts, then move this transaction to *transfers.ledger*. Some banks provide a generic memo for transfers between internal accounts, so it may be necessary to investigate transactions online.
- If the category is incorrect, edit the transaction before moving it from *inbox.ledger*. Changing a category and re-running the import can automatically apply that change to remaining transactions in the inbox.
- To see what categories are available, run *make ui* in order to interactively explore the categories in the ledger.
- If the transaction represents a split transaction, then add lines to the transaction before cut-and-pasting it from *inbox.ledger*.
- If a transaction looks proper, and especially if the booked amount matches a receipt, then the transaction can be immediately marked as "cleared". Uncleared transactions can be investigated later with a special report, after importing is completed.
- When cut-and-pasting, move one or more transaction from the top of the inbox to the bottom of the account ledger you're importing to.

Examples

Purchase Groceries

Let's say you purchase groceries with the Debit card. Ensure the expense is categorized and move that transaction to the ledger for the account: `0123456789.ledger`.

```
2017/01/01 GROCERY
    ; ofxid: 1.00000_0123456789.000000000000000000000000
    Assets:Bank:Debit                       -39.00 CAD
    Expenses:Groceries                       39.00 CAD
```

Move money from Checking to Savings

Let's say you transfer $200 from the Checking account to the Debit account. When that transaction appears in `inbox.ledger`, we process it by moving it to `transfers.ledger`.

```
2017/01/01 TRANSFER OUT
    ; ofxid: 1.00000_0123456789.0000000000000000000000000
    Assets:Bank:Checking                    -200.00 CAD
    Assets:Bank:Debit                        200.00 CAD
```

Automatic Bank Machine

Let's say you withdraw $200 from an ATM that charges a $3 usage fee. Process this transaction by moving it to `transfers.ledger`.

```
2017/01/01 ABM WITHDRAWAL
    ; ofxid: 1.00000_0123456789.0000000000000000000000000
    Assets:Bank:Debit                       -203.00 CAD
    Assets:Cash                              200.00 CAD
    Expenses:Bank                              3.00 CAD
```

Checking the Results

Now see whether the items you just entered are in balance. If everything was entered properly, this will produce a brief balance sheet that only includes your entries. This works under the presumption that you have previously cleared all your other transactions prior to the current work.

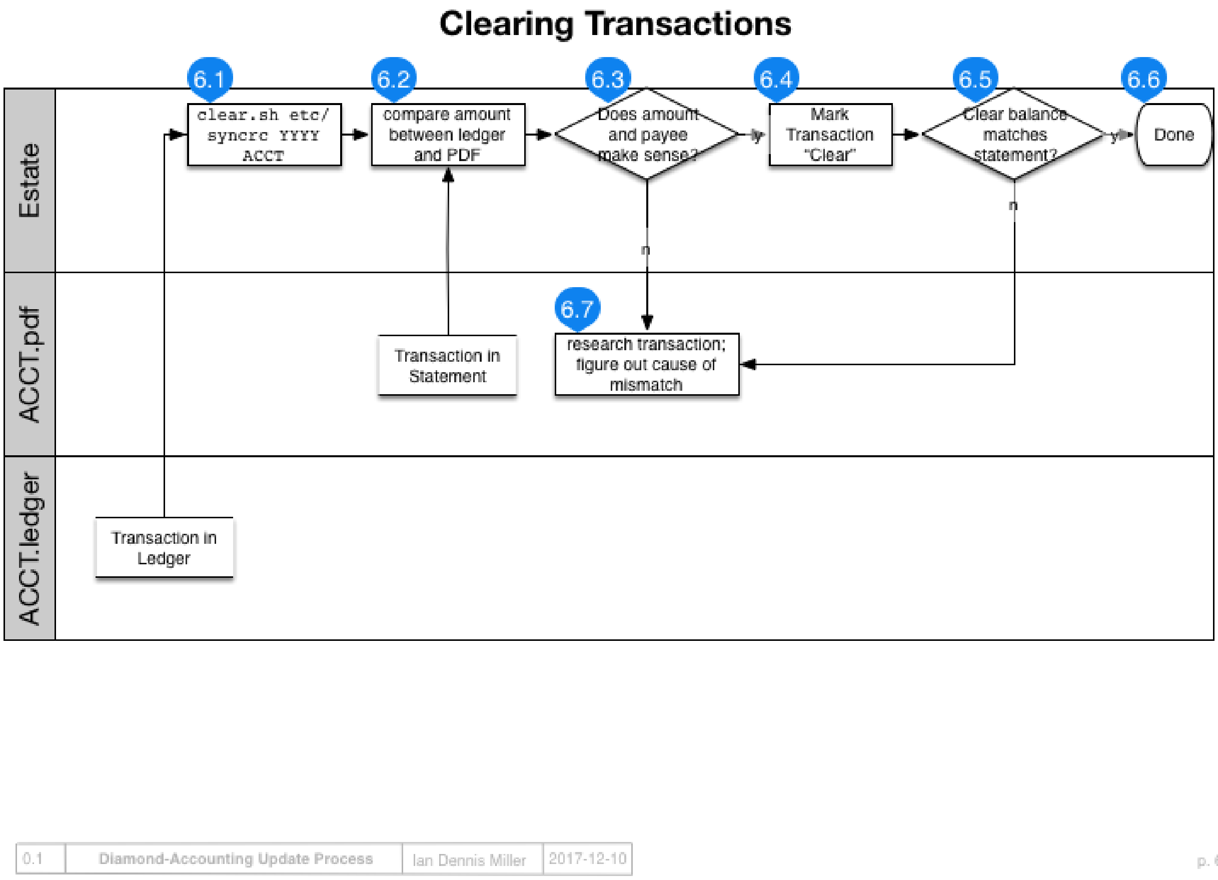
```
make uncleared
```

(or manually run `ledger --uncleared balance`)

Next steps

Next read about *Clearing Transactions* and balances need to match their counterparts as calculated by the bank.

Clearing Transactions



Importing ensures the categories are roughly correct. Now we must verify that the dollar amounts have been imported correctly. Once transactions are imported, they must be reviewed for accuracy once before the "numbers" can be trusted.

The way to review transactions for accuracy is to compare the *cleared* ending balance to a statement from the financial institution. If the balances match, then all of the transactions have been successfully copied over to your ledger; you can now trust the numbers. Transactions must be marked as "cleared" one by one as they are compared between the ledger and the bank statements. This review process is called "clearing transactions."

Each bank statement is like a checkpoint that serves to synchronize our ledger with the bank's ledger. When we independently verify that our record of the transactions sums to the same balance as the bank's, we are confident that the transaction data were downloaded correctly, entered correctly, and are being reported correctly.

Overview

There are a few principles for clearing transactions.

1. open an account ledger and its corresponding bank statement
2. use `clear.sh` to interactively review each uncleared transactions in the account
3. confirm cleared ledger balance matches statement balance

Setup

First ensure you have the following materials in front of you:

- the ledger for the account being cleared, which is in `/ledgers/YYYY/ACCT.ledger`
- the transfers ledger `/ledgers/YYYY/transfers.ledger`
- bank statements in `/statements/`

Clear Transactions

Interactively clear transactions on the command line. To clear account `0123` in the year 2017, use the following command:

```
clear.sh etc/syncrc 2017 0123
```

Ledger supports the ability to mark transactions as cleared by placing an asterisk before the memo. Cleared transactions can be reported apart from uncleared transactions, enabling reports to be generated with accurate numbers even while the ledgers could contain pending transactions.

To clear a transaction, it must be compared against an official record from the party who intermediated the transaction. If you instructed your bank to pay for a coffee, and if this coffee purchase transaction correctly appears in your ledger, then look to the bank statement to confirm that the billed for the coffee shows up. When taking data directly from a bank's exported *QFX* file, it's unlikely that the amounts will differ from the statement. However, when manually entering certain transactions, such as when booking a payment or receivable before it cleared, there is opportunity for data entry errors.

When you use `clear.sh` it relies upon a separate tool called cleartrans-cli. The `cleartrans-cli` tool makes the clearing process operate much more quickly. Although it is simpler to use `clear.sh`, you can manually invoke this on the command line:

```
LEDGER_FILE=~/Work/ledger/ledgers/2017/BANK-0000.ledger cleartrans-cli
ledger --sort date bal --cleared Assets:Bank:Savings
```

Open the statement in one window, open *cleartrans-cli* in another window, and rapidly scan the bank statement for each amount that cycles through the clearing interface.

When a ledger transaction does not appear on the bank statement, it is either 1) in the future; or 2) invalid. Usually, the transaction is in the future and will show up on the following month's statement. In the case of futue transactions, do not clear those transactions: use the interface to skip to the next transaction with the key 'n'. Uncleared transactions will remain in the ledger until next month when they can be cleared against an official statement.

Verify Balances

When `clear.sh` is done running, it will display a balance. This balance should match your statement. If it does, then congratulations: your ledger is probably accurate!

After an entire bank statement's worth of transactions have been cleared, it is time to compare the bottom line. Some statements - in particular, credit cards - like to hide the bottom line or confuse it with other sub-totals. There is only one bottom line that matters - and it **must** match your ledger's bottom line.

You can also manually check the balance of all cleared transactions matches:

```
ledger --sort date bal --cleared Liabilities:CreditCard
```

What to do if the balance doesn't match?

The most common problem is that a transaction was entered but not cleared. Look to *transfers.ledger* and the original ledger being cleared. Calculate the amount you are off by: subtract the bank's balance from the ledger balance. Often times, this amount is a recognizable quantity that will perfectly match exactly one transaction in the ledger. In case the difference looks "familiar," search for that amount and manually mark the transaction as cleared.

Transfers can become convoluted because they involve two accounts you control, meaning it is more likely for these transactions to be double-entered. In that case, the amount of the transfer will be equal to the amount of the difference between the bank statement and the ledger balance. The transaction was probably not cleared, but in case it was actually not entered at all, then manually enter the transfer now.

Sort transactions

The transactions are probably sorted already, so this step may not be necessary. In case transactions are not sorted, the *sorttrans-cli* tool can be used for the job:

```
LEDGER_FILE=~/Work/accounting/ledgers/2017/BANK-0000.ledger sorttrans-cli
```

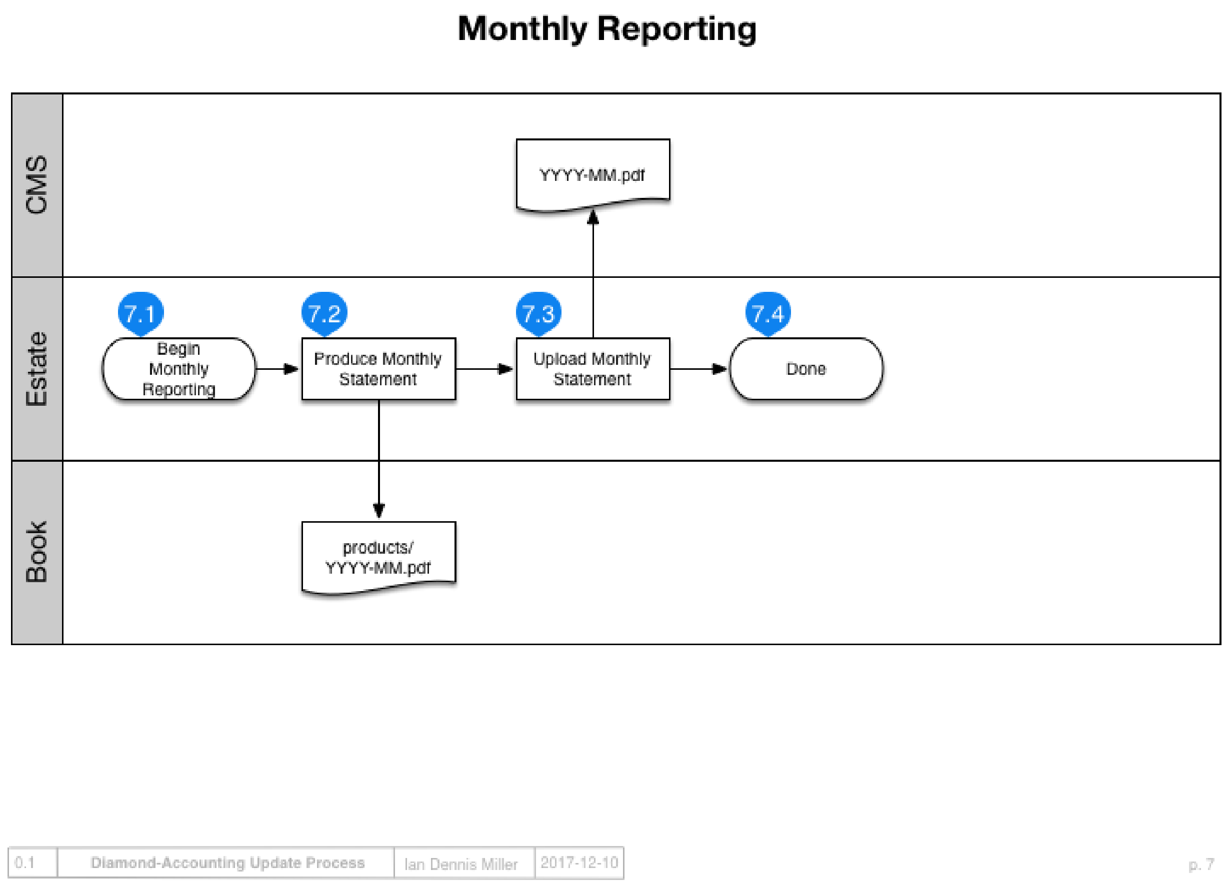
If transactions need to be sorted regularly, then update the *Makefile* with a target so this can be repeated:

```
make sort-0000
```

Next steps

Next read about *Reporting*.

Reporting



The monthly statement is a compilation of common financial statements, including balance, income/expense, and year-to-date graphs. Once the accounts have been imported and cleared, then it is time to create a monthly statement.

Create a Statement

A monthly statement PDF can be generated with *create-statement.sh*.

```
create-statement.sh YYYY MM "NAME ON STATEMENT"
```

For example, to create a statement for January 2017:

```
create-statement.sh 2017 01 "Fancy Pants"
```

The result will be placed into `/products/2017/2017-01.pdf`.

Uploading

Statements should be archived in a CMS for easier use. If you use Joomla for your CMS, then there are two URLs to know:

- http://YOUR.CMS/administrator/index.php?option=com_media&folder=statements
- http://YOUR.CMS/index.php/accounting

To add a statement, first upload the PDF via the Media manager. Once the PDF has been saved to the CMS server, then edit the Accounting page to include a link to the PDF.

- click gear dropdown icon
- click "edit"
- at the bottom of the edit form, click "toggle editor"
- cut-and-paste a link to a previous month and update it for the current month.

NB: multiple files can be uploaded at once.

Next steps

Next read about *Reading the Statement*.

Reading the Statement

This document is a companion to the monthly financial statement. It discusses each sheet of the statement in a general way that should apply to any month.

Balance Sheet

The balance sheet compares assets to liabilities. The "bottom line" of the balance sheet is your equity (also called "net worth"). This is known as the fundamental accounting equation:

```
equity = assets - liabilities
```

You want your equity to be a number above 0.

Budget Balance Sheet

The budget balance sheet compares real expenditures with budget expectations. There are two columns to the budget balance sheet:

- delta, which indicates how close real spending was to expectations
- running balance, which accumulates budget overruns

The sheet is also separated into budgeted expenses as well as unbudgeted. Over time, unbudgeted expenses ought to be minimized, either by budgeting differently or by changing behaviors.

A rule of thumb for budgeting is to keep the balance within 10% of the overall budget. So, a budget of approximately $8000 could vary by $800 (total expenses of $7200 - $8800) and still be considered reasonable. When the budget is outside that range, it loses some usefulness for controlling accounts.

Income and Expense Sheet

The income and expenses sheet summarizes where cash came from, where it went, and how much cashflow there was. The bottom line for this report is the net cashflow. Since expenses are listed as positive numbers, and income as negative, the bottom line for this report ought to be negative in order to maintain healthy finances.

Cashflow Year-to-Date

This plot depicts the monthly rate of cashflow over the course of the year. When income routinely exceeds expenses, a visible wedge will form over the months to indicate the net inflow of cash. If income chronically chases expenses, then you are "living paycheck to paycheck." If income climbs away from expenses, then the accounts are being critically mismanaged and it is time to revisit the budget.

Monthly Income and Expense, Year-to-Date

It can be useful to see how income and expenses vary on a monthly basis over the course of the year. When these amounts are consistent over the course of months, you can infer that your income and expenses are relatively stable. Variability in these amounts could indicate seasonality or lack of financial controls.

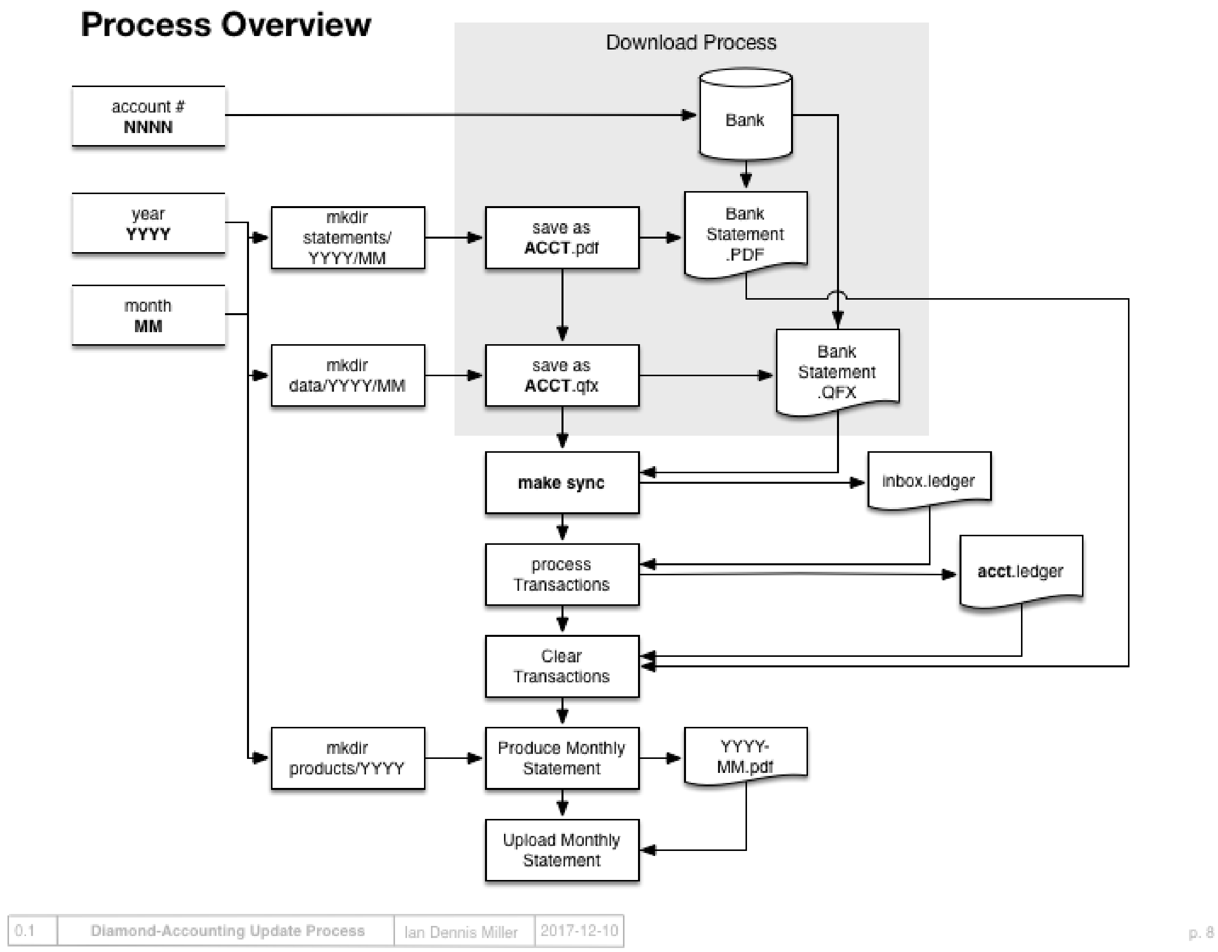
Wealth Growth, Year-to-Date

The "wealthgrow" report is like cashflow for net wealth. Changes in assets and liabilities are plotted over time. There should be relatively little change in liabilities as time goes on - credit needs to be managed and large debts should be paid in predictable increments. Assets that are earning a healthy return will cause the assets slope to be slightly positive.

Next Steps

[To dig deeper, install the ledger project on your workstation.](installation.md)

As long as account transactions can be downloaded in Quicken format, it is possible to import them into a plaintext ledger. The purpose of updating the accounts is to download any transactions that occurred during the previous month, then import all those transactions into the ledger.



1.3.2 Usage

This online documentation can help you get started using *diamond-accounting*.

Configuration

There are two files that are used to control *diamond-accounting*

- syncrc
- ledgerrc

syncrc

Account numbers are difficult to remember so we store this information in `syncrc`. The file called `etc/syncrc` is where you define how to sync from your .QFX files into your ledger accounts. Ledger accounts have plain-language descriptions that are easier to work with than account numbers.

The format of `etc/syncrc` looks like this:

```
[Financial ID] [last 4 digits of account] [ledger account name]
```

The following is an example for an account stored with my first bank. Here is an example for an account called "Checking" with an account number *56781234* - so I will use "1234" as the last 4 digits. The financial ID is 1 because this is my first bank.

```
1 1234 Assets:Bank:Checking
```

Here is longer example example with 3 accounts at my first bank and 1 account at a separate bank:

```
1 1234 Assets:Bank:Checking
1 2345 Assets:Bank:Debit
1 3456 Assets:Bank:Savings
2 0123 Liabilities:CreditCard
```

ledgerrc

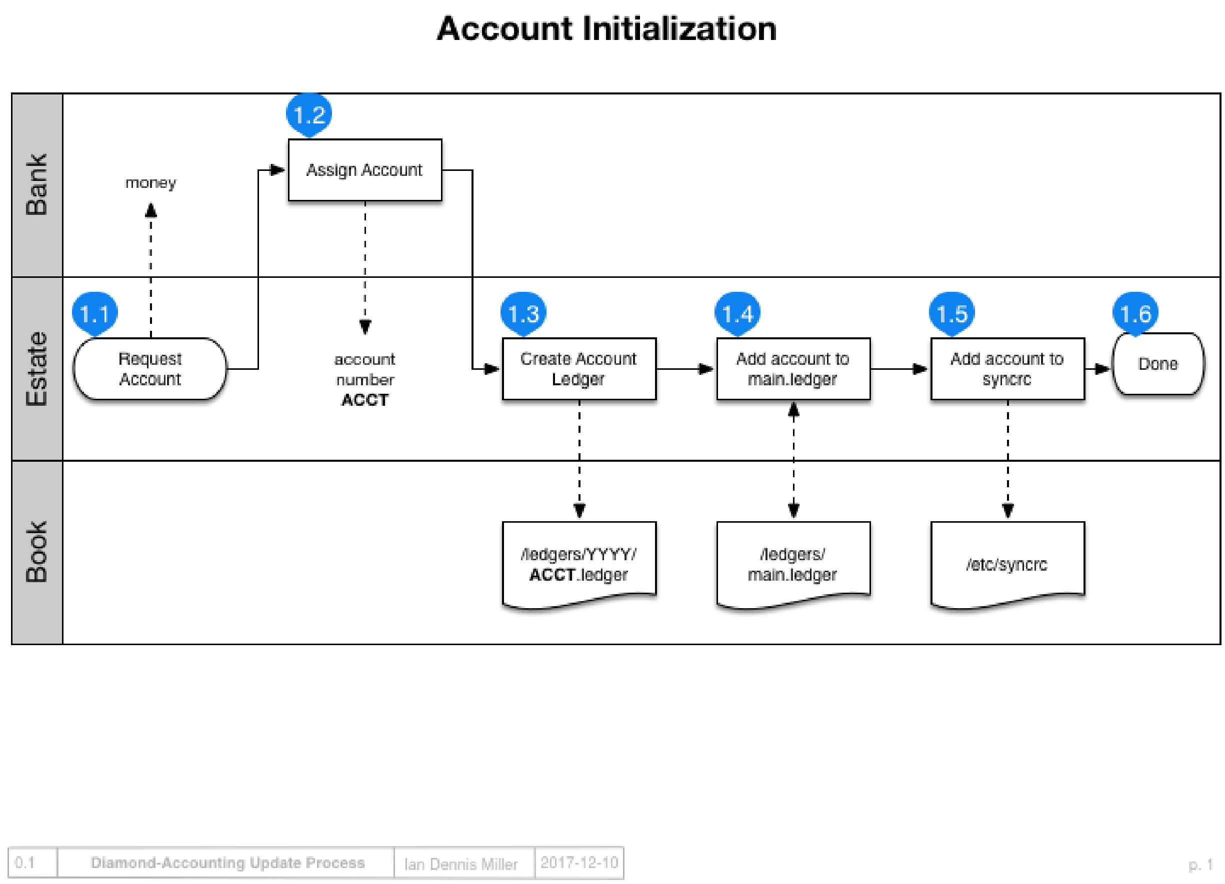
diamond-accounting uses ledger-cli to do the heavy lifting. For ledger to work, this configuration file must point to your actual ledger files.

Install the ledgerrc in your home directory:

```
ln -s etc/ledgerrc ~/.ledgerrc
```

Then, ensure the contents of that file actually point to your `main.ledger`.

Initialization



Organization

I recommend sharing the statements in `/products` with a Content Management System.

Bills

Ledger can help with routine bills in several ways. The most common use I've found is to search transaction memos for specific payees, which is used to look for bill payments.

Queries for Billing

Within the ledger project environment, a variety of ledger "register" operations can zoom in on exactly the desired transactions.

Bills paid by check

```
ledger reg --sort date expenses:rent
ledger reg --sort date expenses:school
```

Automatic Debit Payment

```
ledger reg --sort date @phone
ledger reg --sort date @electricity
ledger reg --sort date @internet
```

Liabilities

```
ledger reg --sort date -r liabilities:card |grep Assets
```

1.3.3 Installation

Typical setup

Python 2.7 is required.

```
pip install diamond-accounting
```

The Python 2.7 dependency comes from ledgerhelpers, which does not yet support Python 3.

Pre-requisites

Ensure python, virtualenv, and ledger are installed. On OS X with homebrew, these can be installed with the following commands:

```
brew install ledger --with-python
brew install python --universal --framework
brew install pyenv-virtualenv pyenv-virtualenvwrapper
```

Diamond-Accounting is not currently compatible with Windows. Sorry.

Python virtualenv

You can optionally make a python virtualenv for accounting work. If you use a virtualenv, then you must include system packages to ensure ledger is available within your environment.

```
mkvirtualenv -a . --system-site-packages accounting
```

Configure

Create and edit a configuration file:

```
ln -s etc/ledgerrc ~/.ledgerrc
```

Ensure `etc/ledgerrc` points to `ledgers/main.ledger`. In my case, that looks like:

```
--file ~/Work/accounting/ledgers/main.ledger
```

Installing meld

This is only relevant for sorting ledger files - and if you never do this, you'll be better off. Sorting is a sensitive process that changes ledger files and should only be run infrequently.

Just install meld from GitHub. The point is to run `meld` on the command line and have it work. The following commands download Meld and install an alias within the virtualenv.

```
wget https://github.com/yousseb/meld/releases/download/osx-9/meldmerge.dmg
open meldmerge.dmg # install it here
ln -s /Applications/Meld.app/Contents/MacOS/Meld ~/.virtualenvs/accounting/bin/meld
```

1.3.4 Reference

Principles

Diamond-Accounting provides an accounting system with the following in mind:

- Store transaction ledgers using plain text files.
- Use Open Source Software and share results with Open Source community.
- Provide a starting file system structure for holding accounting files.
- The software never changes the ledgers. They are essentially write-once.
- Each transaction is written exactly once. Internal transfers between accounts are logged once.
- The raw data produces the ledgers which in turn produce the monthly statements.
- Version control of the ledgers enables simple archival for transactions.
- Produce basic financial statements: balance, income/expense, cashflow.
- Support data entry by multiple people - via version control.

License

Changelog

0.1.0

2017-12-11

- initial release

Artwork

- from Spanish Pictures drawn with pen and pencil.