# LedgerSMB Manual v. 1.3

The LedgerSMB Core Team

June 8, 2012



## Contents

# List of Figures

# Part I
# LedgerSMB and Business Processes

## 1 Introduction to LedgerSMB

### 1.1 What is LedgerSMB

LedgerSMB is an open source financial accounting software program which is rapidly developing additional business management features. Our goal is to provide a robust financial management suite for small to midsize busiensses.

### 1.2 Why LedgerSMB

#### 1.2.1 Advantages of LedgerSMB

- Flexibility and Central Management
- Accessibility over the Internet (for some users)
- Relatively open data format
- Integration with other tools
- Excellent accounting options for Linux users
- Open Source
- Flexible, open framework that can be extended or modified to fit your business.
- Security-conscious development community.

#### 1.2.2 Key Features

- Accounts Receivable
  - Track sales by customer
  - Issue Invoices, Statements, Receipts, and more
  - Do job costing and time entry for customer projects
  - Manage sales orders and quotations
  - Ship items from sales orders
- Accounts Payable
  - Track purchases and debts by vendor
  - Issue RFQ's Purchase Orders, etc.
  - Track items received from purchase orders
- Budgeting
  - Track expenditures and income across multiple departments
  - Track all transactions across departments

- Check Printing
  - Customize template for any check form
- General Ledger
- Inventory Management
  - Track sales and orders of parts
  - Track cost of goods sold using First In/First Out method
  - List all parts below reorder point
  - Track ordering requirements
  - Track, ship, receive, and transfer parts to and from multiple warehouses
- Localization
  - Provide Localized Translations for Part Descriptions
  - Provide Localized Templates for Invoices, Orders, Checks, and more
  - Select language per customer, invoice, order, etc.
- Manufacturing
  - Track cost of goods sold for manufactured goods (assemblies)
  - Create assemblies and stock assemblies, tracking materials on hand
- Multi-company/Multiuser
  - One isolated database per company
  - Users can have localized systems independent of company data set
  - Depending on configuration, users may be granted permission to different companies separately.
- Point of Sale
  - Run multiple cash registers against main LedgerSMB installation
  - Suitable for retail stores and more
  - Credit card processing via TrustCommerce
  - Supports some POS hardware out of the box including:
    * Logic Controls PD3000 pole displays (serial or parallel)
    * Basic text-based receipt printers
    * Keyboard wedge barcode scanners
    * Keyboard wedge magnetic card readers
    * Printer-attached cash drawers
- Price Matrix
  - Track different prices for vendors and customers across the board
  - Provide discounts to groups of customers per item or across the board
  - Store vendors' prices independent of the other last cost in the parts record

- Reporting
  - Supports all basic financial statements
  - Easily display customer history, sales data, and additional information
  - Open framework allows for ODBC connections to be used to generate reports using third party reporting tools.
- Tax
  - Supports Retail Sales Tax and Value Added Tax type systems
  - Flexible framework allows one to customize reports to change the tax reporting framework to meet any local requirement.
  - Group customers and vendors by tax form for easy reporting (1099, EU VAT reporting, and similar)
- Fixed Assets
  - Group fixed assets for easy depreciation and disposal
  - Straight-line depreciation supported out of the box
  - Framework allows for easy development of production-based and time-based depreciation methods
  - Track full or partial disposal of assets

## 1.3 Limitations of LedgerSMB

- No payroll module (Payroll must be done manually)
- Some integration limitations
- Further development/maintenance requires a knowledge of a relatively broad range of technologies

## 1.4 System Requirements of LedgerSMB

- PostgreSQL 8.3 or higher
- A CGI-enabled Web Server (for example, Apache)
- Perl 5.8.x or higher
- An operating system which supports the above software (usually Linux, though Windows, MacOS X, etc. do work)
- LaTeX (optional) is required to create PDF or Postscript invoices
- The following CPAN modules:
  - Data::Dumper
  - Log::Log4perl
  - Locale::Maketext
  - DateTime
  - Locale::Maketext::Lexicon

- DBI
- MIME::Base64
- Digest::MD5
- HTML::Entities
- DBD::Pg
- Math::BigFloat
- IO::File
- Encode
- Locale::Country
- Locale::Language
- Time::Local
- Cwd
- Config::Std
- MIME::Lite
- Template
- Error
- CGI::Simple
- File::MimeInfo

and these optional ones:

- Net::TCLink
- Parse::RecDescent
- Template::Plugin::Latex
- XML::Twig
- Excel::Template::Plus

# 2 Installing LedgerSMB

The process of installing LedgerSMB is described in detail in the INSTALL file which comes with the distribution archive. In the process is:

1. Install the base software: Web Server (Apache), Database server (PostgreSQL) and Perl from your distribution and package manager or source. Read the INSTALL file for details
2. Installing Perl module dependencies from your distribution and package manager or CPAN. Read the INSTALL file for details
3. Give the web server access to the ledgersmb directory
4. Edit ./ledgersmb/ledgersmb.conf to be able to access the database and locate the relevant PostgreSQL contrib scripts
5. Initializing a company database; database setup and upgrade script at http://localhost/ledgersmb/setup.pl
6. Login with your name (database username), password (database user password), Company (databasename)

# 3 User Account and Database Administration Basics

LedgerSMB 1.3 offers a variety of tools for setting up new databases, and most functionality (aside from creating new databases) is now offered dirctly within the main applications. LedgerSMB 1.2 users will note that the admin.pl script is no longer included.

## 3.1 Companies and Datasets

LedgerSMB 1.3 stores data for each company in a separate "database". A database is a PostgreSQL concept for grouping tables, indexes, etc.

To create a database you will need to know a PostgreSQL superuser name and password. If you do not know this information you can set authentication to "trust," then set the password, then set back to a more reasonable setting after this process. Plese see the PostgreSQL documentation for details.

Each company database must be named. This name is essentially the system identifier within PostgreSQL for the company's dataset. The name for the company database can only contain letters, digits and underscores. Additionally, it must start with a letter. Company database names are case insensitive, meaning you can't create two separate company databases called 'Ledgersmb' and 'ledgersmb'.

One way you can create databases fairly easily is by directing your web browser to the setup.pl script at your installed ledgersmb directory. So if the base URL is http://localhost/ledgersmb/, you can access the database setup and upgrade script at http://localhost/ledgersmb/setup.pl. This is very different from the approaches taken by LedgerSMB 1.2.x and earlier and SQL-Ledger, but rather forms a wizard to walk you through the process.

An alternative method is the 'prepare-company-database.sh' script contributed by Erik Huelsmann. This script can be useful in creating and populating databases from the command line and it offers a reference implementation written in BASH for how this process is done.

The 'prepare-company-database.sh' script in the tools/ directory will set up databases to be used for LedgerSMB. The script should be run as 'root' because it wants to 'su' to the postgres user. Alternatively, if you know the password of the postgres user, you can run the script as any other user. You'll be prompted for the password. Additionally, the script creates a superuser to assign ownership of the created company database to. By default this user is called 'ledgersmb'. The reason for this choice is that when removing the ledgersmb user, you'll be told about any unremoved parts of the database, because the owner of an existing database can't be removed until that database is itself removed.

The following invocation of the script sets up your first test company, when invoked as the root user and from the root directory of the LedgerSMB sources:

$ ./tools/prepare-company-database.sh –company testinc

The script assumes your PostgreSQL server runs on 'localhost' with PostgreSQL's default port (5432).

Upon completion, it'll have created a company database with the name 'testinc', a user called 'ledgersmb' (password: 'LEDGERSMBINITIALPASSWORD'), a single user called 'admin' (password: 'admin') and the roles required to manage authorizations.

Additionally, it'll have loaded a minimal list of languages required to succesfully navigate the various screens.

All these can be adjusted using arguments provided to the setup script. See the output generated by the –help option for a full list of options.

## 3.2 How to Create a User

In the database setup workflow, a simple application user will be created. This user, by default, only has user management capabilities. Ideally actual work should be done with accounts which have

fewer permissions.

To set up a user, log in with you administrative credentials (created when initializing the database) and then go to System/Admin Users/Add User. From here you can create a user and add appropriate permissions.

## 3.3 Permissions

Permissions in LedgerSMB 1.3 are enforced using database roles. These are functional categories and represent permissions levels needed to do basic tasks. They are assigned when adding/editing users.

The roles follow a naming convention which allows several LSMB databases to exist on the same PostgreSQL instance. Each role is named lsmb_[dbname]__ followed by the role name. Note that two underscores separate the database and role names. If these are followed then the interface will pick up on defined groups and display them along with other permissions.

### 3.3.1 List of Roles

Roles here are listed minus their prefix (lsmb_[database name]__, note the double underscore at the end of the prefix).

- Contact Management: Customers and Vendors

  **contact_read** Allows the user to read contact information

  **contact_create** Allows the user to enter new contact information

  **contact_edit** Allows the user to update the contact information

  **contact_all** provides permission for all of the above. Member of:
  - contact_read
  - contact_create
  - contact_edit

- Batch Creation and Approval

  **batch_create** Allows the user to create batches

  **batch_post** Allows the user to take existing batches and post them to the books

  **batch_list** Allows the user to list batches and vouchers within a batch. This role also grants access to listing draft tansactions (i.e. non-approved transactions not a part of a batch). Member of:
  - ar_transaction_list
  - ap_transaction_list

- AR: Accounts Receivable

  **ar_transaction_create** Allows user to create transctions. Member of:
  - contact_read

  **ar_transaction_create_voucher** . Allows a user to create AR transaction vouchers. Member of:
  - contact_read
  - batch_create

  **ar_invoice_create** Allows user to create sales invoices. Member of:
  - ar_transaction_create

  **ar_transaction_list** Allows user to view transactions. Member Of:

  - contact_read

**ar_transaction_all** , all non-voucher permissions above, member of:

  - ar_transaction_create
  - ar_transaction_list

**sales_order_create** Allows user to create sales order. Member of:

  - contact_read

**sales_quotation_create** Allows user to create sales quotations. Member of:

  - contact_read

**sales_order_list** Allows user to list sales orders. Member of:

  - contact_read

**sales_quotation_list** Allows a user to list sales quotations. Member of:

  - contact_read

**ar_all** : All AR permissions, member of:

  - ar_voucher_all
  - ar_transaction_all
  - sales_order_create
  - sales_quotation_create
  - sales_order_list
  - sales_quotation_list

- AP: Accounts Payable

**ap_transaction_create** Allows user to create transctions. Member of:

  - contact_read

**ap_transaction_create_voucher** . Allows a user to create AP transaction vouchers. Member of:

  - contact_read
  - batch_create

**ap_invoice_create** Allows user to create vendor invoices. Member of:

  - ap_transaction_create

**ap_transaction_list** Allows user to view transactions. Member Of:

  - contact_read

**ap_transaction_all** , all non-voucher permissions above, member of:

  - ap_transaction_create
  - ap_transaction_list

**purchase_order_create** Allows user to create purchase orders, Member of:

  - contact_read

**rfq_create** Allows user to create requests for quotations. Member of:

  - contact_read

**purchase_order_list** Allows user to list purchase orders. Member of:

  - contact_read

**rfq_list** Allows a user to list requests for quotations. Member of:

  - contact_read

**ap_all** : All AP permissions, member of:

- ap_voucher_all
- ap_transaction_all
- purchase_order_create
- rfq_create
- purchase_order_list
- rfq_list

- Point of Sale

**pos_enter** Allows user to enter point of sale transactions Member of:

- contact_read

**close_till** Allows a user to close his/her till

**list_all_open** Allows the user to enter all open transactions

**pos_cashier** Standard Cashier Permissions. Member of:

- pos_enter
- close_till

**pos_all** Full POS permissions. Member of:

- pos_enter
- close_till
- list_all_open

- Cash Handling

**reconciliation_enter** Allows the user to enter reconciliation reports.

**reconciliation_approve** Allows the user to approve/commit reconciliation reports to the books.

**reconciliation_all** Allows a user to enter and approve reconciliation reports. Don't use if separation of duties is required. Member of:

- reconciliation_enter
- reconciliation_approve

**payment_process** Allows a user to enter payments. Member of:

- ap_transaction_list

**receipt_process** Allows a user to enter receipts. Member of:

- ar_transaction_list

**cash_all** All above cash roles. Member of:

- reconciliation_all
- payment_process
- receipt_process

- Inventory Control

**part_create** Allows user to create new parts.

**part_edit** Allows user to edit parts

**inventory_reports** Allows user to run inventory reports

**pricegroup_create** Allows user to create pricegroups. Member of:

- contact_read

**pricegroup_edit** Allows user to edit pricegroups Member of:

- contact_read

**assembly_stock** Allows user to stock assemblies

**inventory_ship** Allows user to ship inventory. Member of:

- sales_order_list

**inventory_receive** Allows user to receive inventory. Member of:

- purchase_order_list

**inventory_transfer** Allows user to transfer inventory between warehouses.

**warehouse_create** Allows user to create warehouses.

**warehouse_edit** Allows user to edit warehouses.

**inventory_all** All permissions groups in this section. Member of:

- part_create
- part_edit
- inventory_reports
- pricegroup_create
- pricegroup_edit
- assembly_stock
- inventory_ship
- inventory_transfer
- warehouse_create
- warehouse_edit

- GL: General Ledger and General Journal

**gl_transaction_create** Allows a user to create journal entries or drafts.

**gl_voucher_create** Allows a user to create GL vouchers and batches.

**gl_reports** Allows a user to run GL reports, listing all financial transactions in the database. Member of:

- ar_list_transactions
- ap_list_transactions

**yearend_run** Allows a user to run the year-end processes

**gl_all** All GL permissions. Member of:

- gl_transaction_create
- gl_voucher_create
- gl_reports
- yearend_run

- Project Accounting

**project_create** Allows a user to create project entries. User must have contact_read permission to assing them to customers however.

**project_edit** Allows a user to edit a project. User must have contact_read permission to assing them to customers however.

**project_timecard_add** Allows user to add time card. Member of:

- contact_read

**project_timecard_list** Allows a user to list timecards. Necessary for order generation. Member of:

- contact_read

**project_order_generate** Allows a user to generate orders from time cards. Member of:

- project_timecard_list
- orders_generate

- Order Generation, Consolidation, and Management

**orders_generate** Allows a user to generate orders. Member of:

- contact_read

**orders_sales_to_purchase** Allows creation of purchase orders from sales orders. Member of:

- orders_generate

**orders_purchase_consolidate** Allows the user to consolidate purchase orders. Member of:

- orders_generate

**orders_sales_consolidate** Allows user to consolidate sales orders. Member of:

- orders_generate

**orders_manage** Allows full management of orders. Member of:

- project_order_generate
- orders_sales_to_purchase
- orders_purchase_consolidate
- orders_sales_consolidate

- Financial Reports

**financial_reports** Allows a user to run financial reports. Member of:

- gl_reports

- Batch Printing

**print_jobs_list** Allows the user to list print jobs.

**print_jobs** Allows user to print the jobs Member of:

- print_jobs_list

- System Administration

**system_settings_list** Allows the user to list system settings.

**system_settings_change** Allows user to change system settings. Member of:

- system_settings_list

**taxes_set** Allows setting of tax rates and order.

**account_create** Allows creation of accounts.

**account_edit** Allows one to edit accounts.

**auditor** Allows one to access audit trails.

**audit_trail_maintenance** Allows one to truncate audit trails.

**gifi_create** Allows one to add GIFI entries.

**gifi_edit** Allows one to edit GIFI entries.

**account_all** A general group for accounts management. Member of:

- account_create
- account_edit
- taxes_set
- gifi_create
- gifi_edit

**department_create** Allow the user to create departments.

**department_edit** Allows user to edit departments.

**department_all** Create/Edit departments. Member of:

- department_create
- department_edit

**business_type_create** Allow the user to create business types.

**business_type_edit** Allows user to edit business types.

**business_type_all** Create/Edit business types. Member of:

- business_type_create
- business_type_edit

**sic_create** Allow the user to create SIC entries.

**sic_edit** Allows user to edit business types.

**sic_all** Create/Edit business types. Member of:

- sic_create
- sic_edit

**tax_form_save** Allow the user to save the tax form entries.

**template_edit** Allow the user to save new templates. This requires sufficient file system permissions.

**users_manage** Allows an admin to create, edit, or remove users. Member of:

- contact_create
- contact_edit

**system_admin** General role for accounting system administrators. Member of:

- system_setting_change
- account_all
- department_all
- business_type_all
- sic_all
- tax_form_save
- template_edit
- users_manage

- Manual Translation

**language_create** Allow user to create languages

**language_edit** Allow user to update language entries

**part_translation_create** Allow user to create translations of parts to other languages.

**project_translation_create** Allow user to create translations of project descriptions.

**manual_translation_all** Full management of manual translations. Member of:

- language_create
- language_edit
- part_translation_create
- project_translation_create

## 3.4 Creating Custom Groups

Because LedgerSMB uses database roles and naming conventions to manage permissions it is possible to create additional roles and use them to manage groups. There is not currently a way of doing this from the front-end, but as long as you follow the conventions, roles you create can be assigned to users through the front-end. One can also create super-groups that the front-end cannot see but can assign permissions to broups of users on multiple databases. This section will cover both of these approaches.

### 3.4.1 Naming Conventions

In PostgreSQL, roles are global to the instance of the server. This means that a single role can exist and be granted permissions on multiple databases. We therefore have to be careful to avoid naming collisions which could have the effect of granting permissions unintentionally to individuals who are not intended to be application users.

The overall role consists of a prefix and a name. The prefix starts with lsmb_ to identify the role as one created by this application, and then typically the name of the database. This convention can be overridden by setting this in the defaults table (the setting is named 'role_prefix') but this is typically done only when renaming databases. After the prefix follow **two** underscores.

So by default a role for LedgerSMB in a company named mtech_test would start with lsmb_mtech_test__. To create a role for LedgerSMB all we have to do is create one in the database with these conventions.

### 3.4.2 Example

Suppose mtech_test is a database for a financial services company and most users must have appropriate permissions to enter batches etc, but not approve them A role could be created like:

```
CREATE ROLE lsmb_mtech_test__user;
GRANT lsmb_mtech_test__all_ap,
      lsmb_mtech_test__create_batch,
      lsmb_mtech_test__read_contact,
      lsmb_mtech_test__list_batches,
      lsmb_mtech_test__create_contact,
      lsmb_mtech_test__all_gl,
      lsmb_mtech_test__process_payment
TO lsmb_mtech_test__user;
```

Then when going to the user interface to add roles, you will see an entry that says "user" and this can be granted to the user.

# 4 Contact Management

Every business does business with other persons, corporate or natural. They may sell goods and services to these persons or they may purchase goods and services from these persons. With a few exceptions those who are being sold goods and services are tracked as customers, and those from

whom goods and services are being purchased from are vendors. The actual formal distinction is that vendors are entities that the business pays while customers pay the business. Here are some key terms:

**Credit Account** An agreement between your business and another person or business and your business about the payment for the delivery of goods and services on an ongoing basis. These credit accounts define customer and vendor relationships.

**Customer** Another person or business who pays your business money

**Vendor** Another person or business you pay money to.

Prior versions of LedgerSMB required that customers and vendors be entirely separate. In 1.3, however, a given entity can have multiple agreements with the business, some being as a customer, and some being as a vendor.

All customers and vendors are currently tracked as companies, while employees are tracked as natural persons but this will be changing in future versions so that natural persons can be tracked as customers and vendors too.

Each contact must be attached to a country for tax reporting purposes. Credit accounts can then be attached to a tax form for that country (for 1099 reporting in the US or EU VAT reporting).

### 4.1 Addresses

Each contact, whether an employee, customer, or vendor, can have one or more addresses attached, but only one can be a billing address.

### 4.2 Contact Info

Each contact can have any number of contact info records attached. These convey phone, email, instant messenger, etc. info for the individual. New types of records can be generated easily by adding them to the contact_class table.

### 4.3 Bank Accounts

Each contact can have any number of bank accounts attached, but only one can be the primary account for a given credit account. There are only two fields here. The first (BIC, or Banking Institution Code) represents the bank's identifier, such as an ABA routing number, or a SWIFT code, while the second (IBAN) represents the individual's account number.

### 4.4 Notes

In 1.3, any number of read-only notes can be attached either to an entity (in which case they show up for all credit accounts for that entity), or a credit account, in which case they show up only when the relevant credit account is selected.

## 5 Chart of Accounts

The Chart of Accounts provides a basic overview of the logical structure of the accounting program. One can customize this chart to allow for tracking of different sorts of information.

## 5.1 Introduction to Double Entry Bookkeeping

In order to set up your chart of accounts in LedgerSMB you will need to understand a bit about double entry bookkeeping. This section provides a brief overview of the essential concepts. There is a list of references for further reading at the end.

### 5.1.1 Business Entity

You always want to keep your personal expenses and income separate from that of the business or you will not be able to tell how much money it is making (if any). For the same reason you will want to keep track of how much money you put into and take out of the business so you will want to set up a completely seperate set of records for it and treat it almost as if it had a life of its own.

### 5.1.2 Double Entry

Examples:

- When you buy you pay money and receive goods.
- When you sell you get money and give goods.
- When you borrow you get money and give a promise to pay it back.
- When you lend you give money and get a promise to pay it back.
- When you sell on credit you give goods and get a promise to pay.
- When you buy on credit you give a promise to pay and get goods.

You need to record both sides of each transaction: thus double entry. Furthermore, you want to organize your entries, recording those having to do with money in one place, value of goods bought and sold in another, money owed in yet another, etc. Hence you create accounts, and record each half of each transaction in an appropriate account. Of course, you won't have to actually record the amount in more than one place yourself: the program takes care of that.

### 5.1.3 Accounts

**Assets** Money and anything that can be converted into money without reducing the net equity of the business. Assets include money owed, money held, goods available for sale, property, and the like.

**Liabilities** Debts owned by the business such as bank loans and unpaid bills.

**Equity or Capital** What would be left for the owner if all the assets were converted to money and all the liabilities paid off ("Share Capital" on the LedgerSMB default chart of accounts: not to be confused with "Capital Assets".)

**Revenue** Income from business activity: increases Equity

**Expense** The light bill, the cost of goods sold, etc: decreases Equity

All other accounts are subdivisions of these. The relationship between the top-level accounts is often stated in the form of the Accounting Equation (don't worry: you won't have to solve it):

Assets = Liabilities + Equity + (Revenue - Expense)

You won't actually use this equation while doing your bookkeeping, but it's a useful tool for understanding how the system works.

### 5.1.4 Debits and Credits

The words "Debit" and "Credit" come from Latin roots. Debit is related to our word "debt" while credit can indicate a loan or something which edifies an individual or business. The same applies to double entry accounting as it involves equity accounts. Debts debit equity, moneys owed to the business credit the business. Credits to equity accounts make the business more valuable while debits make it less. Thus when you invest money in your business you are crediting that business (in terms of equity), and when you draw money, perhaps to pay yourself, you are debiting that business.

Double entry accounting systems grew out of single entry ones. The goal was to create a system which had inherent checks against human error. Consequently accounts and transactions are arranged such that debits across all accounts always equal credit accounts.

If you invest money in your business that credits an equity account, but the other side of the transaction must thus be a debit. Because the other side of the transaction is an asset account (for example a bank account) it is debited.

Similarly as liability accounts increase, the equity of the business decreases. Consequently, liabilities increase with credits. Income and expense accounts are often the flip sides of transactions involving assets and liailities and represent changes in equity. Therefore they follow the same rules as equity accounts.

- Debits increase assets
- Debits increase expense
- Credits increase liabilities
- Credits increase capital
- Credits increase revenue

Examples:

You go to the bank and make a deposit. The teller tells you that he is going to credit your account. This is correct: your account is money the bank owes you and so is a liability from their point of view. Your deposit increased this liability and so they will credit it. They will make an equal debit to their cash account. When you return you will debit your bank deposits account because you have increased that asset and credit cash on hand because you have decreased that one.

### 5.1.5 Accrual

Early accounting systems were usually run on a cash basis. One generally did not consider money owed to affect the financial health of a company, so expenses posted when paid as did income.

The problem with this approach is that it becomes very difficult or impossible to truly understand the exact nature of the financial health of a business. One cannot get the full picture of the financial health of a business because outstanding debts are not considered. Futhermore, this does not allow for revenue to be tied to cost effectively, so it becomes difficult to assess how profitable a given activity truly is.

To solve this problem, accrual-based systems were designed. The basic principle is that income and expense should be posted as they are incurred, or accrued. This allows one to track income relative to expense for specific projects or operations, and make better decisions about which activities will help one maximize profitability.

To show how these systems differ, imagine that you bill a customer for time and materials for a project you have just completed. The customer pays the bill after 30 days. In a cash based system, you would post the income at the time when the customer pays, while in an accrual system, the income is posted at the time when the project is completed.

### 5.1.6 Separation of Duties

There are two important reasons not to trust accounting staff too much regarding the quality of data that is entered into the system. Human error does occur, and a second set of eyes can help reduce that error considerably.

A second important reason to avoid trusting accounting staff too much is that those with access to financial data are in a position to steal money from the business. All too often, this actually happens. Separation of duties is the standard solution to this problem.

Separation of duties then refers to the process of separating parts of the workflow such that one person's work must be reviewed and approved by someone else. For example, a book keeper might enter transactions and these might later be reviewed by a company accountant or executive. This thus cuts down both on errors (the transaction is not on the books until it is approved), and on the possibility of embezzlement.

Typically, the way duties are likely to be separated will depend on the specific concerns of the company. If fraud is the primary concern, all transactions will be required to go through approval and nobody will ever be allowed to approve their own transactions. If fraud is not a concern, then typically transactions will be entered, stored, and later reviewed/approved by someone different, but allowances may be made allowing someone to review/approve the transactions he/she entered. This latter example doesn't strictly enforce separation of duties, but encourages them nonetheless.

By default, LedgerSMB is set up not to strictly enforce the separation of duties. This can be changed by adding a database constraint to ensure that batches and drafts cannot be approved by the same user that enters them.

In the age of computers, separation of duties finds one more important application: it allows review and approval by a human being of automatically entered transactions. This allows the accounting department to double-check numbers before they are posted to the books and thus avoid posting incorrect numbers (for example, due to a software bug in custom code).

Unapproved transactions may be deleted as they are not full-fledged transactions yet. Approved transactions should be reversed rather than deleted.

Separation of duties is not available yet for sales/vendor invoice documents, but something similar can be handled by feeding them through the order entry workflow (see 12).

### 5.1.7 References

http://www.accounting-and-bookkeeping-tips.com/learning-accounting/accounting-basics-credit.htm
Discussion of debits and credits as well as links to other accounting subjects.

http://www.computer-consulting.com/accttips.htm
Discussion of double entry bookkeeping.

http://www.minnesota.com/~tom/sql-ledger/howtos/
A short glossary, some links, and a FAQ (which makes the "credit=negative number" error). The FAQ focuses on SQL-Ledger, LedgerSMB's ancestor.

http://bitscafe.com/pub2/etp/sql-ledger-notes#expenses
Some notes on using SQL-Ledger (LedgerSMB's ancestor).

http://en.wikipedia.org/wiki/List_of_accounting_topics
Wikipedia articles on accounting.

http://www.bized.ac.uk/learn/accounting/financial/index.htm
Basic accounting tutorial.

http://www.asset-analysis.com/glossary/glo_index.html
Financial dictionary and glossary.

http://www.geocities.com/chapleaucree/educational/FinanceHandbook.html
Financial glossary.

http://www.quickmba.com/accounting/fin/
Explanation of fundamentals of accounting, including good discussions of debits and credits and of double-entry.

### 5.2 General Guidelines on Numbering Accounts

In general, most drop-down boxes in LedgerSMB order the accounts by account number. Therefore by setting appropriate account numbers, one can affect the default values.

A second consideration is to try to keep things under each heading appropriate to that heading. Thus setting an account number for a bank loan account in the assets category is not generally advisable.

If in doubt, review a number of bundled chart of accounts templates to see what sorts of numbering schemes are used.

### 5.3 Adding/Modifying Accounts

These features are listed under System->Chart of Accounts. One can list the accounts and click on the account number to modify them or click on the "add account" option to create new accounts.

- Headings are just broad categories and do not store values themselves, while accounts are used to store the transactional information.
- One cannot have an account that is a summary account (like AR) and also has another function.
- GIFI is mostly of interest to Canadian customers but it can be used to create reports of account hierarchies.

### 5.4 Listing Account Balances and Transactions

One can list the account balances via the Reports->Chart of Accounts report. Clicking on the account number will provide a ledger for that account.

## 6 Administration

This section covers other (non-Chart of Accounts) aspects to the setup of the LedgerSMB accounting package. These are generally accessed in the System submenu.

### 6.1 Taxes, Defaults, and Preferences

Since LedgerSMB 1.2, sales tax has been modular, allowing for different tax accounts to be goverend by different rules for calculating taxes (although only one such module is supplied with LedgerSMB to date). This allows one to install different tax modules and then select which taxes are applied by which programming modules. The sales tax module has access to everything on the submitted form so it is able to make complex determinations on what is taxable based on arbitrary criteria.

The tax rules drop-down box allows one to select any installed tax module (LedgerSMB 1.3 ships only with the simple module), while the ordering is an integer which allows one to specify a tax run which occurs on the form after any rules with lower entries in this box. This allows for compounding of sales tax (for example, when PST applies to the total and GST as well).

As in 1.3.16, new API's have been added to allow one to pass info to tax modules as to minimum and maximum taxable values. These values are used by the Simple tax module as applying to the subtotal of the invoice.

### 6.1.1 Adding A Sales Tax Account

Sales Tax is collected on behalf of a state or national government by the individual store. Thus a sales tax account is a liability– it represents money owed by the business to the government.

To add a sales tax account, create an account in the Chart of Accounts as a liability account, check all of the "tax" checkboxes.

Once this account is created, one can set the tax amount.

### 6.1.2 Setting a Sales Tax Amount

Go to System->Defaults and the tax account will be listed near the bottom of the page. The rate can be set there.

### 6.1.3 Default Account Setup

These accounts are the default accounts for part creation and foreign exchange tracking.

### 6.1.4 Currency Setup

The US accounts list this as USD:CAD:EUR. One can add other currencies in here, such as IDR (Indonesian Rupiah), etc. Currencies are separated by colons.

### 6.1.5 Sequence Settings

These sequences are used to generate user identifiers for quotations, invoices, and the like. If an identifier is not added, the next number will be used.

A common application is to set invoices, etc. to start at 1000 in order to hide the number of issued invoices from a customer.

Leading zeros are preserved. Other special values which can be embedded using <?lsmb ?> tags include:

**DATE** expands to the current date

**YYMMDD** expands to a six-digit version of the date. The components of this date can be re-arranged in any order, so MMDDYY, DDMMYY, or even just MMYY are all options.

**NAME** expands to the name of the customer or vendor

**BUSINESS** expands to the type of business assigned to the customer or ventor.

**DESCRIPTION** expands to the description of the part. Valid only for parts.

**ITEM** expands to the item field. Valid only for parts.

**PERISCOPE** expands to the partsgroup. Valid only for parts.

**PHONE** expands to the telephone number for customers and vendors.

## 6.2 Audit Control

Auditability is a core concern of the architects of any accounting system. Such ensures that any modification to the accounting information leaves a trail which can be followed to determine the nature of the change. Audits can help ensure that the data in the accounting system is meaningful and accurate, and that no foul play (such as embezzlement) is occurring.

### 6.2.1 Explaining transaction reversal

In paper accounting systems, it was necessary to have a means to authoritatively track corrections of mistakes. The means by which this was done was known as "transaction reversal."

When a mistake would be made, one would then reverse the transaction and then enter it in correctly. For example, let us say that an office was renting space for $300 per month. Let us say that they inadvertently entered it in as a $200 expense.

The original transaction would be:

| Account | Debit | Credit |
|---|---|---|
| 5760 Rent | $200 | |
| 2100 Accounts Payable | | $200 |

The reversal would be:

| Account | Debit | Credit |
|---|---|---|
| 5760 Rent | | $200 |
| 2100 Accounts Payable | $200 | |

This would be followed by re-entering the rent data with the correct numbers. This was meant to ensure that one did not erase data from the accounting books (and as such that erasing data would be a sign of foul play).

LedgerSMB has a capability to require such reversals if the business deems this to be necessary. When this option is enabled, existing transactions cannot be modified and one will need to post reversing transactions to void existing transactions before posting corrected ones.

Most accountants prefer this means to other audit trails because it is well proven and understood by them.

### 6.2.2 Close books option

You cannot alter a transaction that was entered before the closing date.

### 6.2.3 Audit Trails

This option stores additional information in the database to help auditors trace individual transactions. The information stored, however, is limited and it is intended to be supplemental to other auditing facilities.

The information added includes which table stored the record, which employee entered the information, which form was used, and what the action was. No direct financial information is included.

## 6.3 Departments

Departments are logical divisions of a business. They allow for budgets to be prepared for the individual department as well as the business as a whole. This allows larger businesses to use LedgerSMB to meet their needs.

#### 6.3.1 Cost v Profit Centers.

In general business units are divided into cost and profit centers. Cost centers are generally regarded as business units where the business expects to lose money and profit centers are where they expect to gain money. For example, the legal department in most companies is a cost center.

One of the serious misunderstandings people run up against is that LedgerSMB tends to more narrowly define cost and profit centers than most businesses do. In LedgerSMB a cost center is any department of the business that does not issue AR transactions. Although many businesses may have cost centers (like technical support) where customer fees may subsidize the cost of providing the service, in LedgerSMB, these are profit centers.

LedgerSMB will not allow cost centers to be associated with AR transactions. So if you want this functionality, you must create the department as a profit center.

### 6.4 Warehouses

LedgerSMB has the ability to track inventory by warehouse. Inventory items can be moved between warehouses, and shipped from any warehouse where the item is in stock. We will explore this concept more later.

### 6.5 Languages

Languages allow for goods and services to be translated so that one can maintain offices in different countries and allow for different goods and service descriptions to be translated to different languages for localization purposes.

### 6.6 Types of Businesses

One can create types of businesses and then give them discounts across the board. For example, one might give a firm that uses one's services as a subcontractor a 10% discount or more.

### 6.7 Misc.

#### 6.7.1 GIFI

GIFI is a requirement for Canadian customers. This feature allows one to link accounts with Canadian tax codes to simplify the reporting process. Some European countries now use a similar system.

People that don't otherwise have a use for GIFI can use it to create reports which agregate accounts together.

#### 6.7.2 SIC

Standard Industrial Classification is a way of tracking the type of business that a vendor or customer is in. For example, an accountant would have an SIC of 8721 while a graphic design firm would have an SIC of 7336. The classification is hierarchical so one could use this field for custom reporting and marketing purposes.

#### 6.7.3 Overview of Template Editing

The templates for invoices, orders, and the like can be edited from within LedgerSMB. The submenus within the System submenu such as HTML Templates, Text Templates and LaTeX templates provide access to this functionality.

### 6.7.4 Year-end

Although the Year-end functionality in LedgerSMB is very useful, it does not entirely make the process simple and painless. One must still manually enter adjustments prior to closing the books. The extent to which these adjustments are necessary for any given business is a matter best discussed with an accountant.

The standard way books are normally closed at the end of the year is by moving all adjusted[1] income and expenses to an equity account usually called 'Retained Earnings.' Assets and liabilities are not moved. Equity drawing/dividend accounts are also moved, but the investment accounts are not. The reasoning behind this process is that one wants a permanent record of the amount invested in a business, but any dividends ought not to count against their recipients when new investors are brought on board.

LedgerSMB automatically moves all income and expense into the specified year-end/retained earnings account. It does not move the drawing account, and this must be done manually, nor does it automate the process of making adjustments.

Contrary to its name, this function can close the books at any time, though this would likely be of limited use.

Once the books are closed, no transactions can be entered into the closed period. Additionally the year end routines cannot be run if there are unapproved transactions in a period to be closed.

## 6.8 Options in the ledger-smb.conf

The ledger-smb.conf configures the software by assigning site-wide variables. Most of these should be left alone unless one knows what one is doing. However, on some systems some options might need to be changed, so all options are presented here for reference:

**auth** is the form of authenticaiton used. If in doubt use 'DB' auth.

**decimal_places** Number of decimal places for money.

**templates** is the directory where the templates are stored.

**sendmail** is the command to use to send a message. It must read the email from standard input.

**language** allows one to set the language for the login screen and admin page.

**latex** tells LedgerSMB whether LaTeX is installed. LaTeX is required for generating Postscript and PDF invoices and the like.

**Environmental variables** can be set here too. One can add paths for searching for LaTeX, etc.

**Printers** section can be used to set a hash table of printers for the software. The primary example is
[printers]
Default = lpr
Color = lpr -PEpson
However, this can use any program that can accept print documents (in Postscript) from standard input, so there are many more possibilities.

**database** provides connection parameters for the database, typically the host and port, but also the location of the contrib scripts (needed for the setup.pl), and the default namespace.

[1] Adjustments would be entered via the General Ledger. The exact process is beyond the scope of this document, however.

# 7 Goods and Services

The Goods and Services module will focus on the definition of goods and services and the related accounting concepts.

## 7.1 Basic Terms

**COGS** is Cost of Goods Sold. When an item is sold, then the expense of its purchase is accrued as attached to the income of the sale.

**List** Price is the recommended retail price.

**Markup** is the percentage increase that is applied to the last cost to get the sell price.

**ROP** is re-order point. Items with fewer in stock than this will show up on short reports.

**Sell** Price is the price at which the item is sold.

## 7.2 The Price Matrix

It is possible to set different prices for different groups of customers, or for different customers individually. Similarly, one can track different prices from different vendors along with the required lead time for an order.

## 7.3 Pricegroups

Pricegroups are used to help determine the discount a given customer may have.

## 7.4 Groups

Groups represent a way of categorizing POS items for a touchscreen environment. It is not fully functional yet, but is sufficient that with some stylesheet changes, it could be made to work.

## 7.5 Labor/Overhead

Labor/overhead is usually used for tracking manufacturing expenses. It is not directly billed to a customer. It is associated with an expense/Cost of Goods Sold (COGS) account.

## 7.6 Services

Services include any labor that is billed directly to the customer. It is associated with an expense/COGS account and an income account. Services can be associated with sales tax.

### 7.6.1 Shipping and Handling as a Service

One approach to dealing with shipping and handling is to add it as a service. Create a service called "Shipping and Handling," with a sell price $1 per unit, and a 0% markup. Bill it as $1 per unit. This allows one to add the exact amount of shipping and handling as necessary.

## 7.7 Parts

A part is any single item you might purchase and either might resell or use in manufacturing an assembly. It is linked to an expense/COGS account, an income account, and an inventory account. Parts can be associated with sales tax.

## 7.8 Assemblies and Manufacturing

Manufacturers order parts but they sell the products of their efforts. LedgerSMB supports manufacturing using the concept of assemblies. An assembly is any product which is manufactured on site. It consists of a selection of parts, services, and/or labor and overhead. Assemblies are treated as parts in most other regards.

However, one cannot order assemblies from vendors. One must instead order the components and stock them once they are manufactured.

### 7.8.1 Stocking Assemblies

One stocks assemblies in the Stock Assembly entry on the Goods and Services submenu. When an assembly is stocked the inventory is adjusted properly.

The Check Inventory option will cause LedgerSMB to refuse to stock an assembly if the inventory required to produce the assembly would drop the part below the reorder point.

## 7.9 Reporting

### 7.9.1 All Items and Parts Reports

The All Items report provides a unified view of assemblies, parts, services, and labor for the company, while the Parts report confines it to parts.

Types of reports are:

**Active** lists all items not marked as obsolete.

**On** Hand lists current inventory .

**Short** Lists all items which are stocked below their ROP.

**Obsolete** Lists all items which are marked as obsolete.

**Orphaned** Lists all items which have never had a transaction associated with them.

One can also list these goods by invoice, order, or quotation.

For best results, it is a good idea to enter some AR and AP data before running these reports.

### 7.9.2 Requirements

This report is designed to assist managers determine the quantities of goods to order and/or stock. It compares the quantity on hand with the activity in a given time frame and provides a list of goods which need to be ordered and the relevant quantity.

### 7.9.3 Services and Labor

This is similar to the Parts and All Items menu but only supports Active, Obsolete, and Orphaned reports.

### 7.9.4 Assemblies

This is similar to the Parts and All Items reports but it also provides an ability to list individual items in the assemblies as well.

AP Invoices, Purchase Orders, and RFQ's are not available on this report.

#### 7.9.5 Groups and Pricegroups

These reports provide a simple interface for locating groups and pricegroups. The report types are similar to what they are for services.

### 7.10 Translations

One can add translations so that they show up in the customer's native language in the issued invoice.

To issue translations, one must have languages defined. One can then add translations to descriptions and part groups.

### 7.11 How Cost of Goods Sold is tracked

Cost of Goods Sold is tracked on a First-In, First-out (FIFO) basis. When a part is purchased, its cost is recorded in the database. The cost of the item is then added to the inventory asset account. When the good is sold, the cost of the item is moved to the cost of goods sold account.

This means that one must actually provide invoices for all goods entered at their actual cost. If one enters in $0 for the cost, the cost of goods sold will also be $0 when the item is sold. We will cover this entire process in more depth after we cover the AP and AR units below.

## 8 Transaction Approval

With the exception of Sales/Vendor Invoices (with inventory control), any financial transaction entered by default must be approved before it shows up in financial reports. Because there are two ways these can be set up, there are two possibly relevant workflows.

For sales/vendor invoices where goods and services are tracked (as distinct from AR/AP transactions which only track amounts), the separation of duties interface is not complete. Here, you should use orders for initial entry, and convert these to invoices.

### 8.1 Batches and Vouchers

Often larger businesses may need to enter batches of transactions which may need to be approved or rolled back as a batch. Batches are thus a generic "container" for vouchers. A given batch can have AR, AP, payment, receipt, and GL vouchers in it together. That same batch, however, will be classified by its main purpose.

For example, one may have a batch for processing payments. That batch may include payment transactions, but also related ar/ap transactions (relating to specific charges relating to payment or receipt). The batch would still be classified as a payment batch however.

In the "Transaction Approval/Batches" screen, one can enter search criteria for batches.

The next screen shows a list of batches including control codes, amounts covered in the batch, and descriptions. Clicking on the control code leads you to a details screen where specific vouchers can be dropped from the batch.

When the batch is approved, all transactions in it are approved with it.

### 8.2 Drafts

Drafts are single transactions which have not yet been approved. For example, a journal entry or AR Transaction would become a "draft" that would need to be approved after entry.

As with batches, one searches for drafts on the first screen and then can approve either on the summary screen or the details screen (by clicking through).

# 9 AP

## 9.1 Basic AP Concepts

The Accounts Payable module tracks all financial commitments that the company makes to other businesses. This includes rent, utilities, etc. as well as orders of goods and services.

## 9.2 Vendors

A vendor is any business that the company agrees to pay money to.

One can enter vendor information under AP->Vendors->Add Vendor. The vendor list can be searched under AP->Vendors->Reports->Search.

Please see the Contact Management section above for more on manageing vendors.

## 9.3 AP Transactions

AP Transactions are generally used for items other than goods and services. Utilities, rent, travel expenses, etc. could be entered in as an AP transaction.

If the item is paid partially or in full when the transaction is entered, one can add payments to the payment section.

All other payments can and should be entered under cash payment (below).

The PO Number and Order Number fields are generally used to track associations with purchase orders sent to vendors, etc. These fields can be helpful for adding misc. expenses to orders for reporting purposes.

The department drop-down box appears when one has created one or more departments. A transaction is not required to be associated with a department, but one can use this feature for budget tracking.

With AP Transactions, there is no option for internal notes. All notes will appear on any printed version of the transaction.

Note: Printing a transaction does not post it. No data is committed until the invoice is posted.

## 9.4 AP Invoices

AP Invoices are used to enter in the receipt of goods and services. Goods and services are deemed entered into the inventory when they are invoiced.

This screen is reasonably similar to the AP Transaction Screen, though the part entry section is a bit different.

The AP Invoice section has a capacity to separate internal notes from notes printed on the invoice. Note, however, that since these are received invoices, it is rare that one needs this ability.

Note that LedgerSMB can search for partial part numbers or descriptions.

Also if you have a group you can use this to select the part.

To remove a line item from an invoice or order, delete the partnumber and click update.

### 9.4.1 Correcting an AP Invoice

If an invoice is entered improperly, the methods used to correct it will vary depending on whether transaction reversal is enforced or not. If transaction reversal is not enforced, one can simply correct the invoice or transaction and repost. Note, however, that this violates generally accepted accounting principles.

If transaction reversal is in effect, one needs to create a duplicate invoice with exactly opposite values entered. If one part was listed as received, then one should enter a negative one for the quantity. Then one can enter the invoice number as the same as the old one. Add an R to the end to show that it is a reversing transaction. Once this is posted, one can enter the invoice correctly.

## 9.5 Cash payment And Check Printing

It is a bad idea to repost invoices/transactions just to enter a payment. The Cash->Payment window allows one to enter payments against AP invoices or transactions.

The printing capability can be used to print checks. The default template is NEBS 9085, though you can use 9082 as well (as Quickbooks does).

The source field is used to store an identifying number of the source document, such as the check number. One must select the item to have it paid, and then enter the amount. One can then print a check.

### 9.5.1 Batch Payment Entry Screen

For bulk payment entry, we provide the batch payment workflow. You can use this to pay any or all vendors filtered by business class or the like. Each payment batch is saved, and hits the books after it is reviewed. It is possible to pay over ten thousand invoices a week using this interface. It is found under Cash/Vouchers/Payments.

## 9.6 Transaction/Invoice Reporting

### 9.6.1 Transactions Report

This report is designed to help you locate AP transactions based on various criteria. One can search by vendor, invoice number, department, and the like. One can even search by the shipping method.

The summary button will show what was placed where, while the details button will show all debits and credits associated with the transaction.

To view the invoice, click on the invoice number. In the detail view, to view the account transactions as a whole, click on the account number.

Open invoices are ones not fully paid off, while closed invoices are those that have been paid.

### 9.6.2 Outstanding Report

The outstanding report is designed to help you locate AP transactions that are not paid yet. The ID field is mostly useful for locating the specific database record if a duplicate invoice number exists.

### 9.6.3 AP Aging Report

This report can tell you how many invoices are past due and by how much.

A summary report just shows vendors while a detail report shows individual invoices.

### 9.6.4 Tax Paid and Non-taxable Report

These reports have known issues. It is better to use the GL reports and filter accordingly.

## 9.7 Vendor Reporting

### 9.7.1 Vendor Search

The Vendor Search screen can be used to locate vendors or AP transactions associated with those vendors.

The basic types of reports are:

**All** Lists all vendors

**Active** Lists those vendors currently active

**Inactive** Lists those vendors who are currently inactive. time frame.

**Orphaned** Lists those vendors who do not have transactions associated with them. These vendors can be deleted.

One can include purchase orders, Requests for Quotations, AP invoices, and AP transactions on this report as well if they occur between the from and to dates.

### 9.7.2 Vendor History

This report can be used to obtain information about the past goods and services ordered or received from vendors. One can find quantities, partnumber, and sell prices on this report. This facility can be used to search RFQ's, Purchase Orders, and AP Invoices.

# 10 AR

## 10.1 Customers

Customers are entered in using the AR->Customers->Add Customer menu.

The salesperson is autopopulated with the current user who is logged in. Otherwise, it looks fairly similar to the Vendor input screen. Customers, like vendors can be assigned languages, but it is more important to do so because invoices will be printed and sent to them.

The credit limit field can be used to assign an amount that one is willing to do for a customer on credit.

### 10.1.1 Customer Price Matrix

The price list button can be used to enter specific discounts to the customer, and groups of customers can be assigned a pricegroup for the purpose of offering specific discounts on specific parts to the customer. Such discounts can be temporary or permanent.

## 10.2 AR Transactions

AR Transactions are where one can add moneys owed the business by customers. One can associate these transactions with income accounts, and add payments if the item is paid when the invoice is issued.

The PO number field is used to track the PO that the customer sent. This makes it easier to find items when a customer is asking for clarification on a bill, for example.

## 10.3 AR Invoices

AR Invoices are designed to provide for the delivery of goods and services to customers. One would normally issue these invoices at the time when the everything has been done that is necessary to get paid by the customer.

As with AP invoices, one can search for matches to partial part numbers and descriptions, and enter initial payments at this screen.

## 10.4 Cash Receipt

The Cash->Receipt screen allows you to accept prepayments from customers or pay single or multiple invoices after they have been posted. One can print a receipt, however the current templates seem to be based on check printing templates and so are unsuitable for this purpose. This presents a great opportunity for improvement.

### 10.4.1 Cash Receipts for multiple customers

The cash->receipts screen allows you to accept payments on all open customer invoices of all customers at once. One could print (directly to a printer only) all receipts to be sent out if this was desired.

## 10.5 AR Transaction Reporting

The AR Outstanding report is almost identical to the AP Outstanding report and is not covered in any detail in this document.

### 10.5.1 AR Transactions Report

This is almost identical to the AP Transactions Report.

If a customer's PO has been associated with this transaction, one can search under this field as well.

### 10.5.2 AR Aging Report

This report is almost identical to the AP Aging report, with the exception that one can print up statements for customer accounts that are overdue. One more application is to calculate interest based on balance owed so that these can be entered as AR transactions associated with the customer.

## 10.6 Customer Reporting

These reports are almost identical to the AP Vendor reports and are not discussed in these notes.

# 11 Projects

## 11.1 Project Basics

A project is a logical collection of AR and AP transactions, orders, and the like that allow one to better manage specific service or product offerings. LedgerSMB does not offer comprehensive project management capabilities, and projects are only used here as they relate to accounting.

One can also add translated descriptions to the project names as well.

## 11.2 Timecards

Timecards allow one to track time entered on specific services. These can then be used to generate invoices for the time entered.

The non-chargeable is the number of hours that are not billed on the invoice.

One can then generate invoices based on this information.

The project field is not optional.

## 11.3 Projects and Invoices

One can select the project id for line items of both AR and AP invoices. These will then be tracked against the project itself.

## 11.4 Reporting

### 11.4.1 Timecard Reporting

The Timecard Report allows one to search for timecards associated with one or more projects. One can then use the total time in issuing invoices (this is not automated yet).

### 11.4.2 Project Transaction Reporting

The Standard or GIFI options can be used to create different reports (for example, for Canadian Tax reporting purposes).

This report brings up a summary that looks sort of like a chart of accounts. Of one clicks on the account numbers, one can see the transactions associated with the project.

### 11.4.3 List of Projects

This provides a simple way of searching for projects to edit or modify.

## 11.5 Possibilities for Using Projects

- One can use them similar to departments for tracking work done for a variety of customers.
- One can use them for customer-specific projects.

# 12 Quotations and Order Management

This unit will introduce the business processes that LedgerSMB allows. These processes are designed to allow various types of businesses to manage their orders and allow for rudimentary customer relationship management processes to be built around this software. In this section, we will introduce the work flow options that many businesses may use in their day-to-day use of the software.

## 12.1 Sales Orders

Sales orders represent orders from customers that have not been delivered or shipped yet. These orders can be for work in the future, for back ordered products, or work in progress. A sales order can be generated form an AR invoice or from a quotation automatically.

## 12.2 Quotations

Quotations are offers made to a customer but to which the customer has not committed to the work. Quotations can be created from Sales orders or AR Invoice automatically.

## 12.3 Shipping

The Shipping module (Shipping->Shipping) allows one to ship portions or entireties of existing sales orders, printing pick lists and packing slips.

One can then generate invoices for those parts that were shipped.

In general, one will be more likely to use these features if they have multiple warehouses that they ship from. More likely most customers will just generate invoices from orders.

## 12.4 AR Work Flow

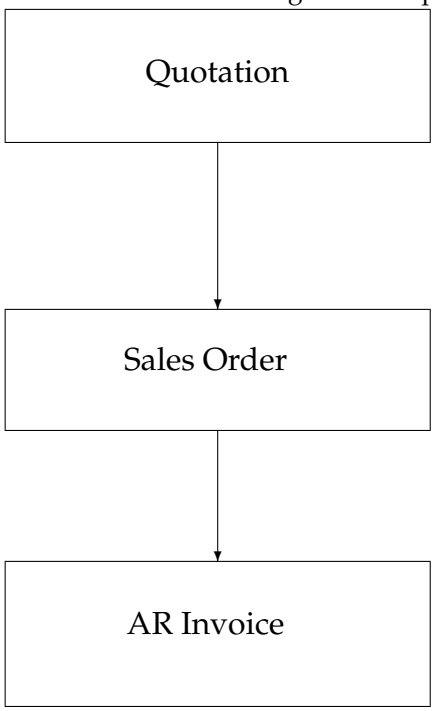
### 12.4.1 Service Example

A customer contacts your firm and asks for a quote on some services. Your company would create a quotation for the job and email it to the customer or print it and mail it. Once the customer agrees to pay, one creates a sales order from the quotation.

When the work is completed, the sales order is converted into a sales invoice and this is presented to the customer as a bill.

Note that in some cases, this procedure may be shortened. If the customer places an order without asking for a quotation and is offered a verbal quote, then one might merely prepare the sales order.

Figure 1: Simple AR Service Invoice Workflow Example



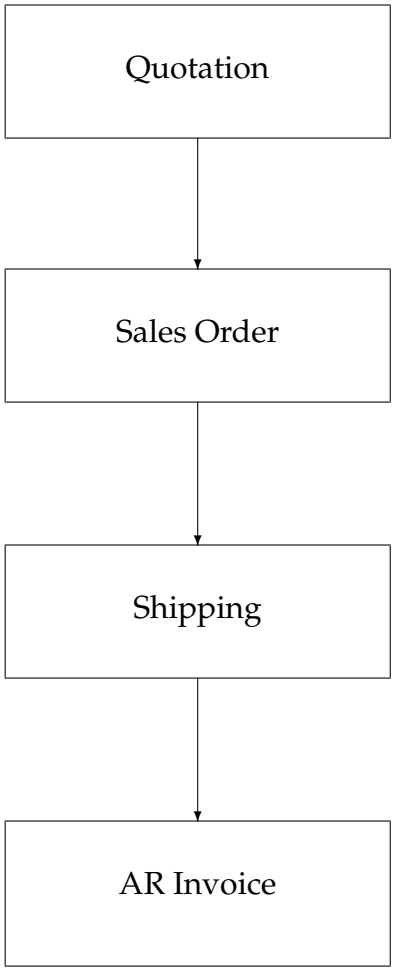
### 12.4.2 Single Warehouse Example

A customer contacts your firm and asks for a quotation for shipping a part. You would create the quotation and when you get confirmation, convert it to an order. Once the parts are in place you could go to shipping and ship the part.

The billing department can then generate the invoice from the sales order based on what merchandise has been shipped and mail it to the customer.

Note that this requires that you have the part in your inventory.

Figure 2: AR Workflow with Shipping

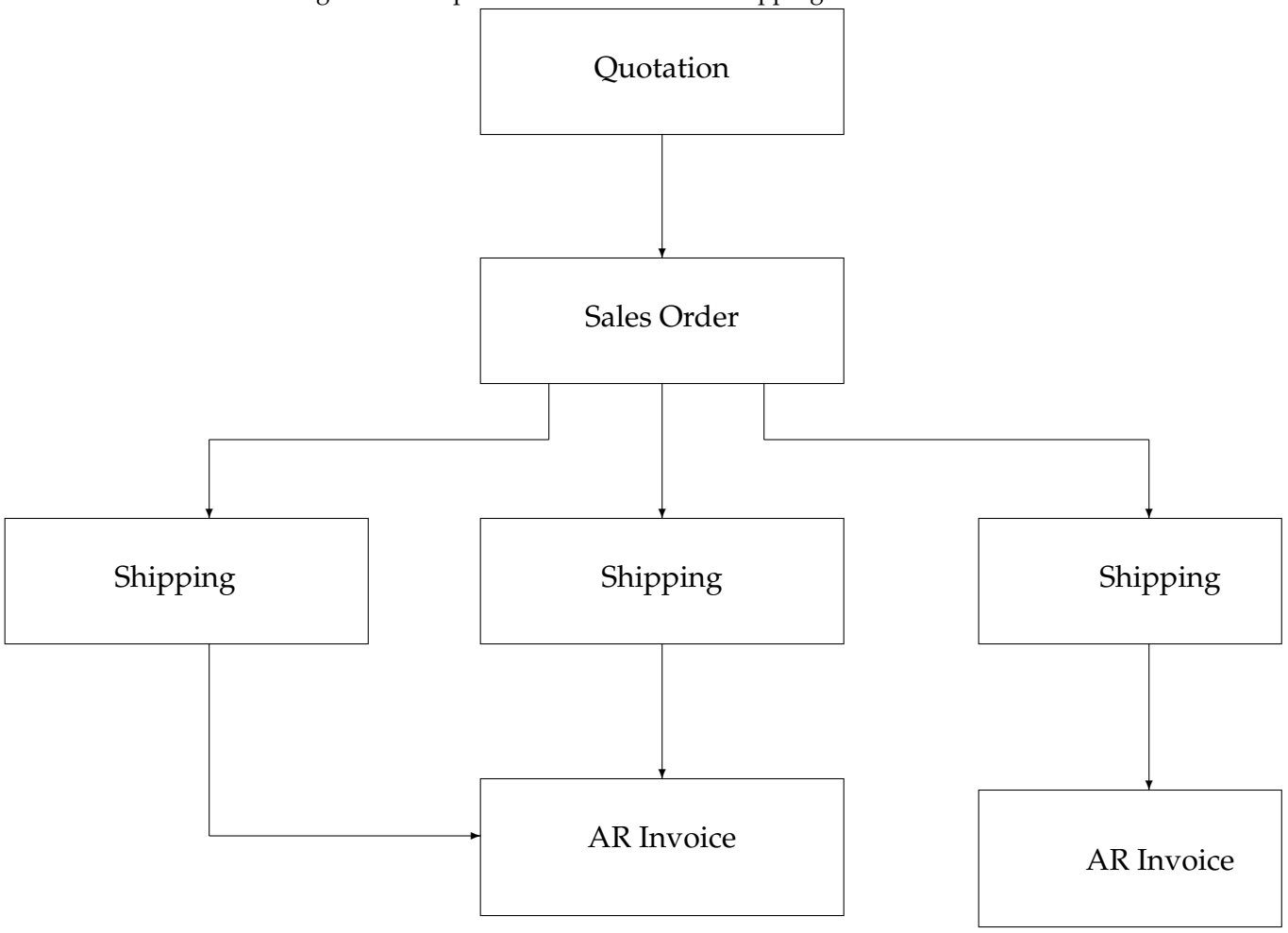



### 12.4.3 Multiple Warehouse Example

A customer contacts your firm and asks for a quotation for a number of different parts. You would create a quotation and when you get confirmation, convert it to a sales order. When you go to ship the item, you would select the warehouse in the drop-down menu, and select the parts to ship. One would repeat with other warehouses until the entire order is shipped.

Then the billing department would go to the sales order and generate the invoice. It would then be mailed to the customer.

Figure 3: Complex AR Workflow with Shipping



## 12.5 Requests for Quotation (RFQ)

A request for quotation would be a formal document one might submit to a vendor to ask for a quote on a product or service they might offer. These can be generated from Purchase Orders or AP Invoices.

## 12.6 Purchase Orders

A purchase order is a confirmation that is issued to the vendor to order the product or service. Many businesses will require a purchase order with certain terms in order to begin work on a product. These can be generated from RFQ's or AP Invoices.

## 12.7 Receiving

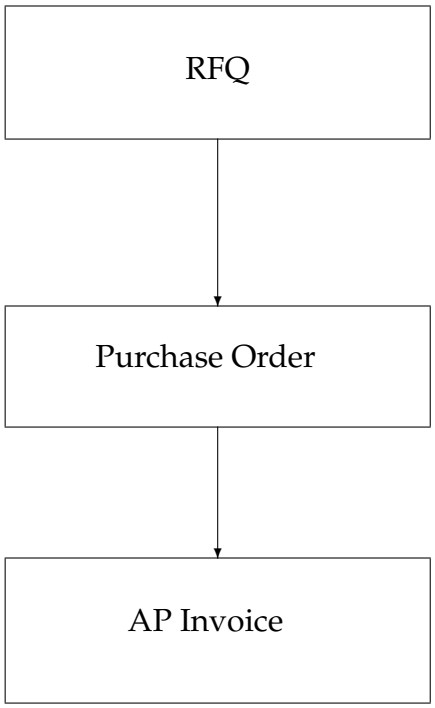
The Shipping->Receiving screen allows you to track the parts received from an existing purchase order. Like shipping, it does not post an invoice but tracks the received parts in the order.

## 12.8 AP Work Flow

### 12.8.1 Bookkeeper entering the received items, order completed in full

Your company inquires about the price of a given good or service from another firm. You submit an RFQ to the vendor, and finding that the price is reasonable, you convert it to an order, adjust the price to what they have quoted, and save it. When the goods are delivered you convert the order into an AP invoice and post it.

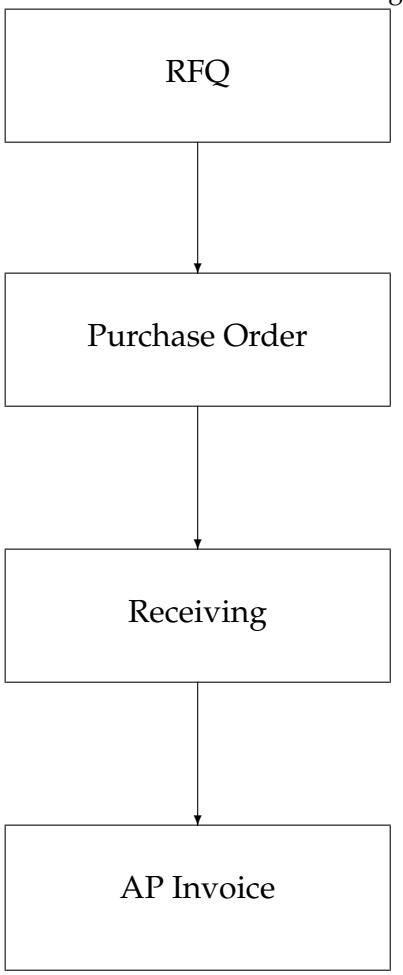
Figure 4: Simple AP Workflow



### 12.8.2 Bookkeeper entering received items, order completed in part

Your company inquires about the price of a given good or service from another firm, You submit an RFQ to the vendor, and finding that the price is acceptable, you convert it into an order, adjusting the price to what they have quoted, and save it. When some of the goods are received, you open up the purchase order, enter the number of parts received, convert that order into an invoice, and post it. Repeat until all parts are received.

Figure 5: AP Workflow with Receiving

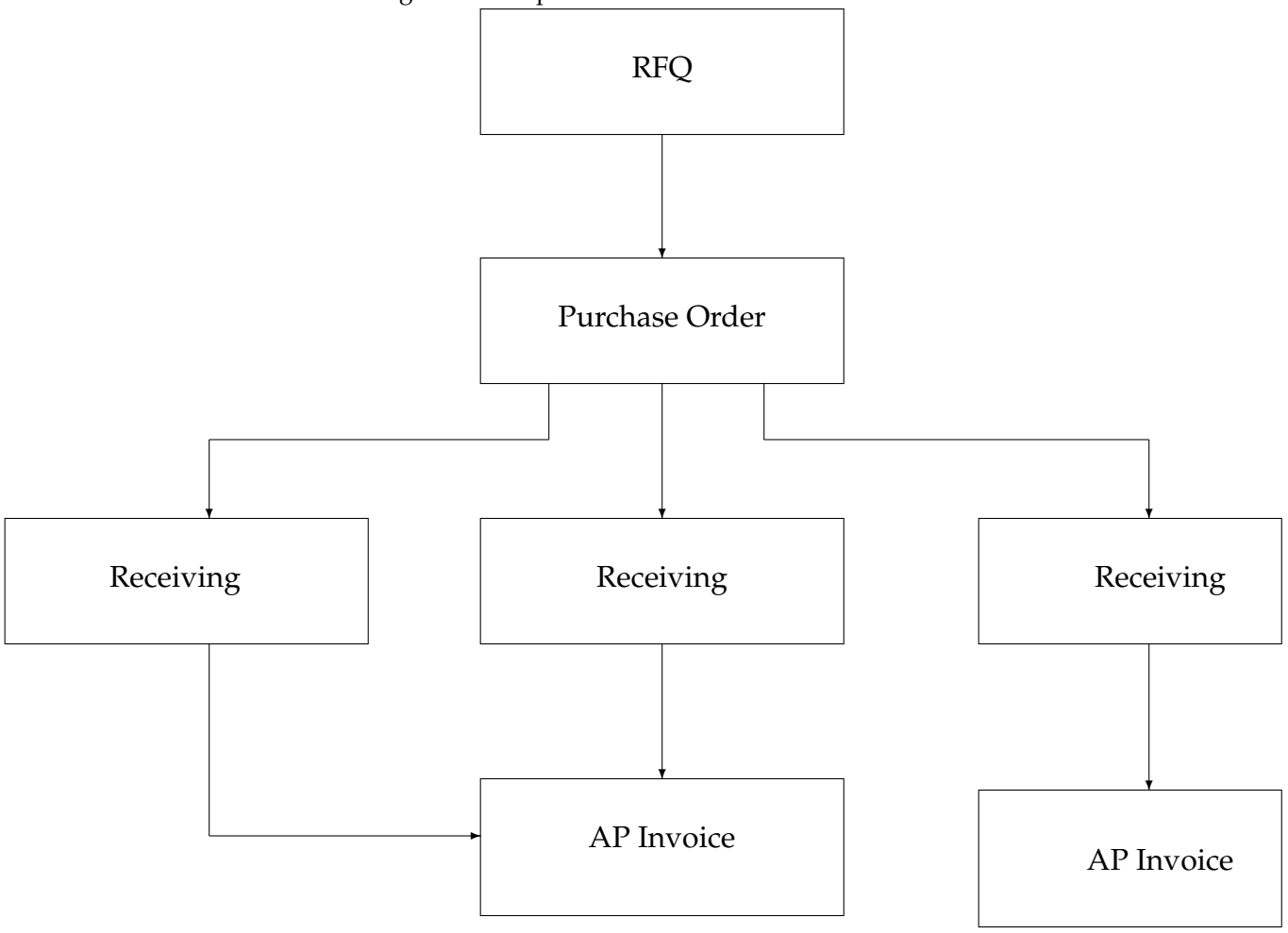



### 12.8.3 Receiving staff entering items

Your company inquires about the price of a given good or service from another firm, You submit an RFQ to the vendor, and finding that the price is acceptable, you convert it into an order, adjusting the price to what they have quoted, and save it. When some or all of the goods are received, the receiving staff goes to Shipping-Receiving, locates the purchase order, and fills in the number of items received.

The bookkeeper can then determine when all items have been received and post the invoice at that time.

Figure 6: Complex AP Workflow



## 12.9 Generation and Consolidation

### 12.9.1 Generation

The Generation screen allows you to generate Purchase Orders based on sales orders. One selects the sales orders one wants to use, and clicks "Generate Purchase Orders." Then one selects clicks on the parts to order, adjusts the quantity if necessary, and clicks "Select Vendor." This process is repeated for every vendor required. Then the Generate Orders button is clicked.

### 12.9.2 Consolidation

One can consolidate sales and/or purchase orders using this screen. For the consolidation to work you must have more than one order associated with the relevant customer or vendor.

## 12.10 Reporting

The reporting functionality in the order management is largely limited to the ability to locate purchase orders, sales orders, RFQ's, and quotations.

## 12.11 Shipping Module: Transferring Inventory between Warehouses

One can transfer inventory between warehouses if necessary by using the Shipping->Transfer Inventory screen.

# 13 Fixed Assets

One of the new features in LedgerSMB 1.3.x is fixed asset management, which includes tracking, depreciation, and disposal. In general, the LedgerSMB approach to these topics is implemented in a streamlined fashion but with an ability to add advanced methods of depreciation later.

## 13.1 Concepts and Workflows

Fixed asset management and accounting provides a better ability to track the distribution of expenses relative to income. Many fixed assets may be depreciated so that the expense of obtaining the asset can be spread across the usable life of that asset. This is done in accordance with the idea of attempting to match actual resource consumption (as expenses) with income in order to better gauge the financial health of the business.

### 13.1.1 Fixed Assets and Capital Expenses

Fixed assets are pieces of property that a business may own which cannot be easily converted into cash. They are differentiated from liquid assets, which include inventory, cash, bank account balances, some securities, etc. Fixed assets, by their nature are typically purchased and used for an extended period of time, called the estimated usable life. During the estimated usable life, the fixed asset is being utilized by the business, and so we would want to track the expense as gradually incurred relative to the income that the asset helps produce. This expense is called a "capital expense" and refers to either a purchase of a fixed asset or an expense which improves it.

Examples of capital expenses and fixed assets might include (using a pizza place as an example):

- A company vehicle
- Tables, chairs, ovens, etc.

- Changes to the leased property needed for the business.
- Major repairs to the company vehicle.

### 13.1.2 Asset Classes

LedgerSMB allows assets and capital expenses to be grouped together in asset classes for easy management. An asset class is a collection of assets which are depreciated together, and which are depreciated in the same way. One cannot mix depreciation methods within a class. Account links for the asset class are used as default links for fixed assets, but assets may be tied to different accounts within an asset class.

### 13.1.3 Depreciation

Depreciation is a method for matching the portion of a capital expense to income related to it. Expenses are depreciated so that they are spread out over the usable life of the asset or capital expense. Depreciation may be linear or non-linear and maybe spread out over time or over units of production. For example, one might wish to depreciate a car based on miles driven, over a usable life of, say, 100000 miles, or one might want to depreciate it based on a useful life of five years.

LedgerSMB currently only supports variations on straight-line depreciation, either with an estimated life measured in months or in years.

Depreciation is subject to separation of duties. Permissions for entering and approving depreciation reports are separate, and the GL transactions created must currently be approved in order to show up on the books.

### 13.1.4 Disposal

Fixed assets may be disposed of through sale or abandonment. Abandonment generally is a method of giving the asset away. A sale involves getting something for the asset.

The disposal workflow is conceptually similar to the depreciation workflow except that additionally one can track proceeds for sales, and tie in gains/losses to appropriate income or expense accounts.

Gains are realized where the salvage value is greater than the undepreciated value of the asset, and losses where the salvage value is less.

### 13.1.5 Net Book Value

Net book value represents the value to depreciate of fixed assets. It is defined as the basis value minus depreciation that has been recorded. The basis is further defined as the purchase value minus the estimated salvage value for LedgerSMB purposes. We track all capital expenses separately for depreciation purposes, and so capital expenses which adjust value of other fixed assets have their own net book value records. This is separate from how capital gain and loss might need to be tracked for tax purposes in the US.

### 13.1.6 Supported Depreciation Methods

Currently we only ship with the following variations on the straight line depreciation method:

**Annual Straight Line Daily** Life is measured in years, depreciation is an equal amount per year, divided up into equal portions daily.

**Annual Straight Line Monthly** Life is measured in years, depreciation is an equal amount per year, divided up into equal portions each month. This differs from daily in that February would have

less depreciation than August. This module is more commonly used than the daily depreciation because it is easier to calculate and thus more transparent.

**Whole Month Straight Line** Life is measured in months, and depreciation occurs only per whole month.

# 14 HR

The HR module is currently limited to tracking employees for and their start and end dates. It has very little other functionality. One could build payroll systems that could integrate with it however.

# 15 POS

LedgerSMB 1.2 includes a number of components merged from Metatron Technology Consulting's SL-POS. Although it is still not a perfect solution, it is greatly improved in both workflow and hardware support. It is suitable for retail establishments at the moment.

## 15.1 Sales Screen

The sales screen looks very much like a normal invoice entry screen with a few differences.

- The discount text field is not available, nor is the unit field..
- The next part number is automatically focused when the data loads for rapid data entry.
- Hot keys for the buttons are Alt-U for update, Alt-P for print, Alt-O for post, and Alt-R for print and post.
- Part Groups appear at the bottom of the screen.
- Alt-N moves the cursor to the next free payment line.

## 15.2 Possibilities for Data Entry

- Barcode scanners can be used to scan items in as they are being rung in.
- One could use touch screens, though this would ideally require some custom stylesheets to make it efficient.

## 15.3 Hardware Support

As LedgerSMB is a web-based application, the web browser usually does not allow the page to write to arbitrary files. Therefore hardware support for pole displays, etc. is not readily possible from the application itself. LedgerSMB gets around this limitation by using an additional set of network sockets from the server to the client to control its hardware. This naturally requires that other software is also running on the client.

Notes for specific types of hardware are as follows:

**Touch** screens: The default stylesheet is not really usable from a touchscreen as the items are often too small. One would need to modify the stylesheets to ensure that the relevant items would be reasonable. Setting down the resolution would also help.

**Receipt** Printers: ESC/POS printers generally work in text mode. Control sequences can be embedded in the template as necessary.

**Pole** Displays: Generally supported. Only the Logic Controls PD3000 is supported out of the box, but making this work for other models ought to be trivial.

**Cash** Drawers: These should be attached to the printer. The control codes is then specified in the pos.conf.pl so that the command is sent to the printer when the open till button is pushed.

**Barcode** Scanners: Most customers use decoded barcode scanners through a keyboard wedge interface. This allows them to scan items as if they were typing them on the keyboard.

## 15.4 Reports

### 15.4.1 Open Invoices

The POS->Open screen allows one to find any POS receipts that are not entirely paid off.

### 15.4.2 Receipts

The POS->Receipts screen allows one to bring up a basic record of the POS terminals. It is not sufficient for closing the till, however, though it may help for reconciliation.

The till column is the last component or octet of the terminal's IP address. Therefore it is a good idea to try to avoid having IP addresses where the last octet is the same.

All entries are grouped by date and source in this report.

# 16 General Ledger

## 16.1 GL Basics

The General Ledger is the heart of LedgerSMB. Indeed, LedgerSMB is designed to be as close as possible to a software equivalent of a paper-based accounting program (but with no difference between the General Ledger and General Journal).

### 16.1.1 Paper-based accounting systems and the GL

In order to understand the principle of the General Ledger, one must have a basic understanding of the general process of bookkeeping using double-entry paper-based accounting systems.

Normally when a transaction would be recorded, it would first be recorded in the "General Journal" which would contain detailed information about the transaction, notes, etc. Then the entries from the General Journal would be transcribed to the General Ledger, where one could keep closer tabs on what was going on in each account.

In the general journal, all transactions are listed chronologically with whatever commentary is deemed necessary, while in the general ledger each account has its own page and transactions are recorded in a simple and terse manner. The General Journal is the first place the transaction is recorded and the General Ledger is the last.

At the end of the accounting period, the GL transactions would be summarized into a trial balance and this would be used for creating financial statements and closing the books at the end of the year.

### 16.1.2 Double Entry Examples on Paper

Let us say that John starts his business with an initial investment of $10,000.

This is recorded in the General Journal as follows (in this example, suppose it is page 1):

| Date | Accounts and Explanation | Ref | DEBIT | CREDIT |
|---|---|---|---|---|
| March 1 | Checking Account<br>John Doe Capital<br>John Doe began a business with an investment of $10000 | 1060<br>3011 | 10000.00 | <br>10000.00 |

This would then be transcribed into two pages of the General Ledger. The first page might be the Checking Account page:

| DATE | EXPLANATION | REF. | DEBITS | DATE | EXPLANATION | REF. | CREDITS |
|---|---|---|---|---|---|---|---|
| March 1 | | J1 | 10000.00 | | | | |

On the John Doe Capital page, we would add a similar entry:

| DATE | EXPLANATION | REF. | DEBITS | DATE | EXPLANATION | REF. | CREDITS |
|---|---|---|---|---|---|---|---|
| | | | | March 1 | | J1 | 10000.00 |

### 16.1.3 The GL in LedgerSMB

The paper-based accounting procedure works well when one is stuck with paper recording requirements but it has one serious deficiency— all of this transcribing creates an opportunity for errors.

Relational databases relieve the need for such transcription as it is possible to store everything physically in a way similar to the way a General Journal is used in the paper-based systems and then present the same information in ways which are more closely related to the General Ledger book.

This is the exact way that the General Ledger is used in LedgerSMB. The actual data is entered and stored as if it was a general journal, and then the data can be presented in any number of different ways.

All modules of LedgerSMB that involve COA accounts store their data in the General Ledger (it is a little more complex than this but this is very close to the actual mechanism).

## 16.2 Cash Transfer

The simplest form of GL entry in LedgerSMB is the Cash->Transfer screen. This screen shows two transaction lines, and fields for reference, department, description, and notes.

The field descriptions are as follows:

**Reference** refers to the source document for the transfer. One can use transfer sheets, bank receipt numbers, etc for this field.

**Description** is optional but really should be filled in. It ought to be a description of the transaction.

**Notes** provide supplemental information for the transaction.

**FX** indicates whether foreign exchange is a factor in this transaction.

**Debit** indicates money going **into** the asset account.

**Credit** indicates money coming **out** of the asset account.

**Source** is the source document for that portion of the transaction.

**Memo** lists additional information as necessary.

**Project** allows you to assign this line to a project.

The credit and debit options seem to be the opposite of what one would think of concerning one's bank account. The reason is that your bank statement is done from the bank's point of view. Your bank account balance is an asset to you and therefor you show it as having a debit balance, but to the bank it is money they owe you and so they show it as having a credit balance.

Note that in this screen, when an item is updated, it will reduce the number of lines to those already filled in plus an extra line for the new line in the data entry.

## 16.3 GL Transactions

The GL Transaction screen (General Ledger->Add Transaction) is identical to the Cash Transfer screen with the exception that it starts with nine lines instead of two. Otherwise, they are identical.

Again, one must be careful with debits and credits. Often it is easy to get confused. It is generally worth while to go back to the principle that one tracks them with regard to their impact on the equity accounts. So expenses are credits because they debit the equity accounts, and income is a debit because it credits the retained earning equity account.

## 16.4 Payroll as a GL transaction

Currently payroll must be done as a GL transaction. The attempts to create a payroll system that would ship with LSMB have largely stalled.

Most customers running their businesses will have an idea of how to do this.

Figure 7: Payroll as a GL Transaction (Purely fictitious numbers)

| Account | Debit | Credit |
|---|---|---|
| 5101 Wages and Salaries | 500 | |
| 2032 Accrued Wages | | 450 |
| 2033 Fed. Income Tax wthd | | 30 |
| 2034 State Inc. Tax. wthd | | 15 |
| 2035 Social Security wthd | | 3 |
| 2036 Medicare wthd | | 2 |
| 2032 Accrued Wages | 450 | |
| 1060 Checking Acct | | 450 |

## 16.5 Reconciliation

To reconcile an account (say, when one would get a checking account statement), one would go to cash/reconciliation, and check off the items that have cleared. One can then attempt to determine where any errors lie by comparing the total on the statement with the total that LSMB generates.

This can be done for other accounts too, such as petty cash.[2] Some users even reconcile liability accounts and the likes.

In LedgerSMB 1.3.x, the reconciliation framework has been completely rewritten to allow for easier reconciliation especially where there are large numbers of transactions. It is now possible to reconcile accounts with thousands of transactions per month, and to track what was reconciled in each report. Reconciliation is now also subject to separation of duties, allowing a reconciliation report to be committed to the books only when approved.

The reconciliation screen is now divided into four basic parts. At the top is a header with information about which account is being reconcilied, ending balances, variances, etc. Then comes a list of cleared transactions, as well as totals of debits and credits.

[2]Petty cash denotes a drawer of cash that is used to pay small expenses. When an expense is paid, it is recorded on a slip of paper that is stored for reconciliation purposes.

After this comes a (usually empty) list of failed matches when the file import functionality is used. These again list the source, then the debits/credits in LedgerSMB and the debits/credits in the bank import.

Finally there is a list of uncleared transactions. To move error a transaction from the error or uncleared section into the cleared section, check one or more off and click update. Update also checks for new uncleared transactions in the reconciliation period, and it saves the current report so it can be continued later.

### 16.5.1 File Import Feature

1.3.x has a plugin model that allows one to write parsers against a variety of file formats, one or more per account. The file can be placed in the LedgerSMB/Reconciliation/CSV/Formats directory, and the function must be called parse_ followed by the account id. The function must be in the LedgerSMB::Reconciliation::CSV namespace.

This obviously has a few limitations. Hosting providers might want to start the account tables 10000 apart as far as the initial id's go or provide dedicated LedgerSMB instances per client.

## 16.6 Reports

The most flexible report in LedgerSMB is the GL report because it has access to the entire set of financial transactions of a business. Every invoice posted, payment made or received, etc. can be located here.

The search criteria include:

**Reference** is the invoice number, or other reference number associated with the transaction.

**Source** is the field related to the source document number in a payment or other transaction.[3]

**Memo** relates to the memo field on a payment.

**Department** can be used to filter results by department.

**Account** Type can be used to filter results by type of account (Asset, Liability, etc.)

**Description** can be used to filter by GL description or by customer/vendor name.

The actual format of the report looks more like what one would expect in a paper accounting system's general journal than a general ledger per se. A presentation of the data that is more like the paper general ledger is found in the Chart of Accounts report.

### 16.6.1 GL as access to almost everything else

The GL reports can be used to do all manner of things. One can determine, for example, which AP invoice or transaction was paid with a certain check number or which invoice by a specific customer was paid by a specific check number.

# 17 Recurring Transactions

Any transaction or invoice may be repeated a number of times in regular intervals. To schedule any GL, AR, or AP transaction or invoice, click the schedule button.

[3] Source documents are things like receipts, canceled checks, etc. that can be used to verify the existence and nature of a transaction.

In general the reference number should be left blank as this will force LedgerSMB to create a new invoice or transaction number for each iteration. The rest of the options are self-explanatory. Note that a blank number if iterations will result in no recurrences of the transaction.

To process the recurring transactions, click on the Recurring Transactions option on the main menu select the ones you want to process and click "Process Transactions."

An enhanced recurring transaction interface is forthcoming from the LedgerSMB project.

# 18 Financial Statements and Reports

Financial statements and reports are a very important part of any accounting system. Accountants and business people rely on these reports to determine the financial soundness of the business and its prospects for the next accounting period.

## 18.1 Cash v. Accrual Basis

Financial statements, such as the Income Statement and Balance Sheet can be prepared either on a cash or accrual basis. In cash-basis accounting, the income is deemed earned when the customer pays it, and the expenses are deemed incurred when the business pays them.

There are a number of problems with cash-basis accounting from a business point of view. The most serious is that one can misrepresent the wellbeing of a business by paying a large expense after a deadline. Thus cash-basis accounting does not allow one to accurately pair the income with the related expense as these are recorded at different times. If one cannot accurately pair the income with the related expense, then financial statements cannot be guaranteed to tell one much of anything about the well-being of the business.

In accrual basis accounting, income is considered earned when the invoice is posted, and expenses are considered incurred at the time when the goods or services are delivered to the business. This way, one can pair the income made from the sale of a product with the expense incurred in bringing that product to sale. This pairing allows for greater confidence in business reporting.

## 18.2 Viewing the Chart of Accounts and Transactions

The Reports–>Chart of Accounts will provide the chart of accounts along with current totals in each account.

If you click on an account number, you will get a screen that allows you to filter out transactions in that account by various criteria. One can also include AR/AP, and Subtotal in the report.

The report format is similar to that of a paper-based general ledger.

## 18.3 Trial Balance

### 18.3.1 The Paper-based function of a Trial Balance

In paper-based accounting systems, the accountant at the end of the year would total up the debits and credits in every account and transfer them onto another sheet called the trial balance. The accountant would check to determine that the total debits and credits were equal and would then transfer this information onto the financial statements. It was called a trial balance because it was the main step at which the error-detection capabilities of double-entry accounting systems were used.

### 18.3.2 Running the Trial Balance Report

This report is located under Reports –>Trial Balance. One can filter out items by date, accounting period, or department. One can run the report by accounts or using GIFI classifications to group accounts together.

From this report, you can click on the account number and see all transactions on the trial balance as well as whether or not they have been reconciled.

### 18.3.3 What if the Trial Balance doesn't Balance?

If the trial balance does not balance, get technical support immediately. This usually means that transactions were not entered properly. Some may have been out of balance, or some may have gone into non-existent accounts (believe it or not, LedgerSMB does not check this latter issue).

### 18.3.4 Trial Balance as a Summary of Account Activity

The trial balance offers a glance at the total activity in every account. It can provide a useful look at financial activity at a glance for the entire business.

### 18.3.5 Trial Balance as a Budget Planning Tool

By filtering out departments, one can determine what a department earned and spent during a given financial interval. This can be used in preparing budgets for the next accounting period.

## 18.4 Income Statement

The Income Statement is another tool that can be used to assist with budgetary planning as well as provide information on the financial health of a business.

The report is run from Reports–>Income Statement. The report preparation screen shows the following fields:

**Department** allows you to run reports for individual departments. This is useful for budgetary purposes.

**Project** allows you to run reports on individual projects. This can show how profitable a given project was during a given time period.

**From** and To allow you to select arbitrary from and to dates.

**Period** allows you to specify a standard accounting period.

**Compare to** fields allow you to run a second report for comparison purposes for a separate range of dates or accounting period.

**Decimalplaces** allows you to display numbers to a given precision.

**Method** allows you to select between accrual and cash basis reports.

**Include** in Report provides various options for reporting.

**Accounts** allows you to run GIFI reports instead of the standard ones.

The report shows all income and expense accounts with activity during the period when the report is run, the balances accrued during the period, as well as the total income and expense at the bottom of each section. The total expense is subtracted from the total income to provide the net income during the period. If there is a loss, it appears in parentheses.

### 18.4.1 Uses of an Income Statement

The income statement provides a basic snapshot of the overall ability of the business to make money. It is one of the basic accounting statements and is required, for example, on many SEC forms for publicly traded firms.

Additionally, businessmen use the income statement to look at overall trends in the ability of the business to make money. One can compare a given month, quarter, or year with a year prior to look for trends so that one can make adjustments in order to maximize profit.

Finally, these reports can be used to provide a look at each department's performance and their ability to work within their budget. One can compare a department or project's performance to a year prior and look for patterns that can indicate problems or opportunities that need to be addressed.

## 18.5 Balance Sheet

The balance sheet is the second major accounting statement supported by LedgerSMB. The balance sheet provides a snapshot of the current financial health of the business by comparing assets, liabilities, and equity.

In essence the balance sheet is a statement of the current state of owner equity. Traditionally, it does not track changes in owner equity in the same way the Statement of Owner Equity does.

The Balance Sheet report preparation screen is much simpler than the Income Statement screen. Balance sheets don't apply to projects, but they do apply to departments. Also, unlike an income statement, a balance sheet is fixed for a specific date in time. Therefore one does not need to select a period.

The fields in creating a balance sheet are:

**Department** allows you to run separate balance sheets for each department.

**As** at specifies the date. If blank this will be the current date.

**Compare to** specifies the date to compare the balance sheet to.

**Decimalplaces** specifies the number of decimal places to use.

**Method** selects between cash and accrual basis.

**Include** in report allows you to select supplemental information on the report.

**Accounts** allows you to select between standard and GIFI reports.

The balance sheet lists all asset, liability, and equity accounts with a balance. Each category has a total listed, and the total of the equity and liability accounts is also listed.

The total assets should be equal to the sum of the totals of the liability and equity accounts.

## 18.6 What if the Balance Sheet doesn't balance?

Get technical support immediately, This may indicate that out of balance transactions were entered or that transactions did not post properly.

## 18.7 No Statement of Owner Equity?

The Statement of Owner Equity is the one accounting statement that LedgerSMB does not support. However, it can be simulated by running a balance sheet at the end of the time frame in question and comparing it to the beginning. One can check this against an income statement for the period in question to verify its accuracy. The statement of owner equity is not as commonly used now as it once was.

# 19 The Template System

LedgerSMB allows most documents to be generated according to a template system. This allows financial statements, invoices, orders, and the like to be customized to meet the needs of most businesses. Company logos can be inserted, the format can be radically altered, one can print letters to be included with checks to vendors instead of the checks themselves, and the like. In the end, there is very little that cannot be accomplished regarding modification of these documents with the template system.

One can define different templates for different languages, so that a customer in Spain gets a different invoice than a customer in Canada.

LedgerSMB provides templates in a variety of formats including text, html, LaTeX, Excel, and Open Document Spreadsheat. Each of these is processed using TemplateToolkit[4] with a few specific modifications:

- start tag is <?lsmb and end tag is ?>
- text(string) is available as a function in the templates for localization purposes.
- gettext(string, language) can be used to translate a string into a specified language rather than the default language (useful for multi-lingual templates).

Additionally the UI/lib/ui-header.html can be used to provide standardized initial segments for HTML documents and inputs can be entered via interfaces in the UI/lib/elements.html template.

### 19.0.1 What is LaTeX ?

LaTeX (pronounced LAY-tech) is an extension on the TeX typesetting system. It largely consists of a set of macros that allow one to focus on the structure of the document while letting the TeX engine do the heavy lifting in terms of determining the optimal formatting for the page. LaTeX is used in a large number of academic journals (including those of the American Mathematics Association). It is available at http://www.tug.org and is included in most Linux distributions.

Like HTML, LaTeX uses plain text documents to store the formatting information and then when the document is rendered, attempts to fit it onto a page. LaTeX supports the concept of stylesheets, allowing one to separate content from format, and this feature is used in many higher-end applications, like journal publication.

Unlike HTML, LaTeX is a complete though simple programming language that allows one to redefine internals of the system for formatting purposes.

This document is written in LaTeX.

### 19.0.2 Using LyX to Edit LaTeX Templates

LyX is a synchronous LaTeX editor that runs on Windows, UNIX/Linux, and Mac OS X. It requires an installed LaTeX-2e implementation and can be obtained at http://www.lyx.org. Like the most common LaTeX implementations, it is open source and is included with most Linux distributions.

## 19.1 Customizing Logos

LaTeX requires different formats of logos depending on whether the document is going to be generated as a PDF or as postscript. Postscript requires an embedded postscript graphic, while PDF requires any type of graphic other than embedded postscript. Usually one uses a PNG's for PDF's, though GIF's could be used as well. The logo for a LaTeX document must be fully qualified as to its path.

[4] See http://template-toolkit.org/ for more information and documentation.

HTML documents can have logos in many different formats. PNG's are generally preferred for printing reasons. The image can be stored anywhere and merely referenced in the HTML.

Note: Always test the an invoice with images to ensure that the rest of the page format is not thrown off by it.

### 19.2 How are They Stored in the Filesystem?

The template directory ("templates" in the root LedgerSMB install directory) contains all the root templates used by LedgerSMB. The default templates are stored in the demo folder.

Inside the templates directory are one or more subdirectories where the relevant templates have been copied as default language templates for the user. Many users can use the same user directory (which bears the name of the LedgerSMB username). Within this directory are more subdirectories for translated templates, one for each language created. If the requested language template is not found, LedgerSMB will attempt to translate the one in the main folder.

### 19.3 Upgrade Issues

When LedgerSMB is upgraded, the templates are not replaced. This is designed to prevent the upgrade script from overwriting changes made during the course of customizing the templates.

Occasionally, however, the data model changes in a way which can cause the templates to stop printing certain information. When information that was showing up before an upgrade stops showing up, one can either upgrade the templates by copying the source template over the existing one, or one can edit the template to make the change.

## 20 An Introduction to the CLI for Old Code

This section applies to the following Perl scripts:

**am.pl** System Management

**ap.pl** Accounts Payable Transactions and Reports

**ar.pl** Accounts Receivable Transactions and Reports

**bp.pl** Batch Printing

**ca.pl** Chart of Accounts (some functions, others are in the accounts.pl script)

**gl.pl** General Ledger Reports and Journal Entry

**ic.pl** Inventory control

**ir.pl** Invoices Received (vendor invoices)

**is.pl** Invoices Shipped (sales invoices)

**jc.pl** Job costing (timecards)

**oe.pl** Order Entry

**pe.pl** Projects

**ps.pl** Point of Sale

**rp.pl** Reports

## 20.1 Conventions

The command-line API will be referred to as the API.

## 20.2 Preliminaries

All scripts included in the documentation can also be found in the doc/samples directory.

Consider a simple example:

cd /usr/local/ledger-smb ./ct.pl "login=name&path=bin&password=xxxxx&action=search&db=customer"

The cd command moves your terminal session's current working directory into the main LedgerSMB directory. Then the LedgerSMB perl script ct.pl is called with one long line as an argument. The argument is really several variable=value pairs separated by ampersands (&). The value for the login variable is the username that LedgerSMB is to use, and the value for the password variable is the plaintext password.

To build our examples we will use a username of "clarkkent" who has a password of "lOis,lAn3".

cd /usr/local/ledger-smb ./ct.pl "login=clarkkent&path=bin&password=lOis,lAn3&action=search&db=customer"

If we execute these commands we will get the html for the search form for the customer database. This result isn't useful in itself, but it shows we are on the right track.

## 20.3 First Script: lsmb01-cli-example.sh

With a working example, we can start to build reproducible routines that we can grow to do some useful work.

This is a bash script which:

1. sets NOW to the current working directory
2. prompts for and reads your LedgerSMB login
3. prompts for and reads (non-echoing) your LedgerSMB password
4. changes directory to /usr/local/ledger-smb
5. constructs login and logout commands and a transaction command
6. logins into ledger-smb (in a real program, output would be checked for success or failure)
7. executes the transaction
8. logs out of ledger-smb (although this is not necessary)
9. returns to the original working directory
10. exits

Running lsmb01-cli-example.sh produces:

$ lsmb01-cli-example.sh

LedgerSMB login: clarkkent

LedgerSMB password:

```
<body>

<form method=post action=ct.pl>

<input type=hidden name=db value=customer>
```

```
<table width=100%>
  <tr>
    <th class=listtop>Search</th>
.
.
.
```

A script like this would work well for simple batch transactions, but bash is not a very friendly language for application programming.

A nicer solution would be to use a language such as perl to drive the command line API.

### 20.3.1 Script 1 (Bash)

```
#!/bin/bash
######################################################################
#
# lsmb01-cli-example.sh
# Copyright (C) 2006. Louis B. Moore
#
# $Id: $
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
#
######################################################################

NOW=`pwd`

echo -n "Ledger-SMB login: "
read LSLOGIN
echo

echo -n "Ledger-SMB password: "
stty -echo
read LSPWD
stty echo
echo

ARG="login=${LSLOGIN}&password=${LSPWD}&path=bin&action=search&db=customer"

LGIN="login=${LSLOGIN}&password=${LSPWD}&path=bin&action=login"
LGOT="login=${LSLOGIN}&password=${LSPWD}&path=bin&action=logout"
```

```
cd /usr/local/ledger-smb

./login.pl $LGIN 2>&1  > /dev/null
./ct.pl $ARG
./login.pl $LGOT 2>&1  > /dev/null

cd $NOW

exit 0
```

## 20.4 Second Script: lsmb02-cli-example.pl

Our second script is written in perl and logs you in but it still uses the API in its simplest form, that is, it builds commands and then executes them. This type of script can be used for more complex solutions than the simple bash script above, though it is still fairly limited. If your needs require, rather than have the script build and then execute the commands it could be written to generate a shell script which is executed by hand.

This script begins by prompting for your LedgerSMB login and password. Using the supplied values a login command is constructed and passed into the runLScmd subroutine. runLScmd changes directory to /usr/local/ledger-smb/ for the length of the subroutine. It formats the command and executes it and returns both the output and error information to the caller in a scalar.

The script checks to see if there was an error in the login, exiting if there was.

Next, the script reads some records which are stored in the program following the __END__ token. It takes each record in turn, formats it then feeds each transaction through runLScmd and looks at the results for a string that signals success.

Once all the transactions are processed, runLScmd is called one last time to logout and the script exits.

### 20.4.1 Script 2 (Perl)

```
#!/usr/bin/perl -w
#
#  File:         lsmb02-cli-example.pl
#  Environment:  Ledger-SMB 1.2.0+
#  Author:       Louis B. Moore
#
#  Copyright (C)   2006  Louis B. Moore
#
#  This program is free software; you can redistribute it and/or
#  modify it under the terms of the GNU General Public License
#  as published by the Free Software Foundation; either version 2
#  of the License, or (at your option) any later version.
#
#  This program is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.
#
#  You should have received a copy of the GNU General Public License
#  along with this program; if not, write to the Free Software
#  Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
```

```
#
#  Revision:
#       $Id$
#
#

use File::chdir;
use HTML::Entities;


print "\n\nLedger-SMB login: ";
my $login = <STDIN>;
chomp($login);


print "\nLedger-SMB password: ";
system("stty -echo");
my $pwd = <STDIN>;
system("stty echo");
chomp($pwd);
print "\n\n";

$cmd = "login=" . $login . '&password=' . $pwd . '&path=bin&action=login';

$signin = runLScmd("./login.pl",$cmd);

if ( $signin =~ m/Error:/ ) {

print "\nLogin error\n";
exit;

}


while (<main::DATA>) {

chomp;
@rec = split(/\|/);

$arg = 'path=bin/mozilla&login=' . $login . '&password=' . $pwd .
'&action='       . escape(substr($rec[0],0,35)) .
'&db='           . $rec[1] .
'&name='         . escape(substr($rec[2],0,35)) .
'&vendornumber=' . $rec[3] .
'&address1='     . escape(substr($rec[4],0,35)) .
'&address2='     . escape(substr($rec[5],0,35)) .
'&city='         . escape(substr($rec[6],0,35)) .
'&state='        . escape(substr($rec[7],0,35)) .
'&zipcode='      . escape(substr($rec[8],0,35)) .
'&country='      . escape(substr($rec[9],0,35)) .
'&phone='        . escape(substr($rec[10],0,20)) .
'&tax_2150=1' .
```

```
'&taxaccounts=2150' .
'&taxincluded=0' .
'&terms=0';

$rc=runLScmd("./ct.pl",$arg);

if ($rc =~ m/Vendor saved!/) {

print "$rec[2] SAVED\n";

} else {

print "$rec[2] ERROR\n";

}

}


$cmd = "login=" . $login . '&password=' . $pwd . '&path=bin&action=logout';

$signin = runLScmd("./login.pl",$cmd);

if ( $signin =~ m/Error:/ ) {

    print "\nLogout error\n";

}

exit;


#*****************************************************
# Subroutines
#*****************************************************


sub runLScmd {

    my $cmd  = shift;
    my $args = shift;
    my $i    = 0;
    my $results;

    local $CWD = "/usr/local/ledger-smb/";

    $cmd = $cmd . " \"" . $args . "\"";

    $results = `$cmd 2>&1`;

    return $results;
```

```
}

sub escape {

    my $str = shift;

    if ($str) {

decode_entities($str);
$str =~ s/([^a-zA-Z0-9_.-])/sprintf("%%%02x", ord($1))/ge;
    }

    return $str;

}


#*****************************************************
# Record Format
#*****************************************************
#
# action | db | name | vendornumber | address1 | address2 | city | state | zipcode | cou
#

__END__
save|vendor|Parts are Us|1377|238 Riverview|Suite 11|Cheese Head|WI|56743|USA|555-123-33
save|vendor|Widget Heaven|1378|41 S. Riparian Way||Show Me|MO|39793|USA|555-231-3309|
save|vendor|Consolidated Spackle|1379|1010 Binary Lane|Dept 1101|Beverly Hills|CA|90210|
```

# Part II
# Technical Overview

## 21 Basic Architecture

LedgerSMB is a web-based Perl program that interfaces with PostgreSQL using the relevant Perl modules. The code is well partitioned, and the main operation modules are written in an object oriented way.

### 21.1 The Software Stack

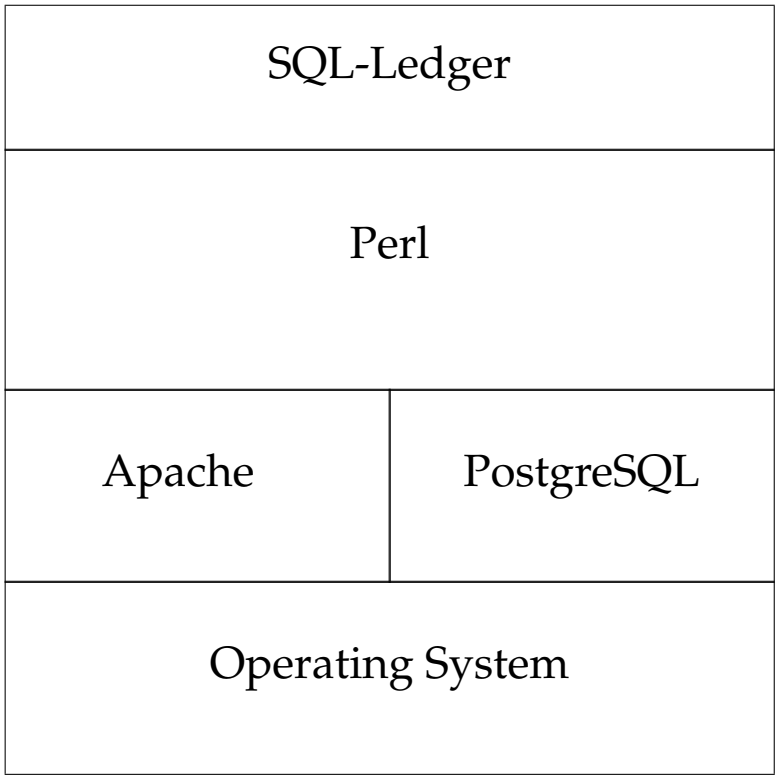



Figure 8: The LedgerSMB software stack in a Typical Implementation

LedgerSMB runs in a Perl interpreter. I do not currently know if it is possible to run it with Perl2C or other language converters to run in other environments. However, except for high-capacity environments, Perl is a good language choice for this sort of program.

LedgerSMB used to support DB2 and Oracle as well as PostgreSQL. However, currently some of the functionality is implemented using PostgreSQL user-defined functions. These would need to be ported to other database managers in order to make the software work on these. It should not be too hard, but the fact that it has not been done yet may mean that there is no real demand for running the software under other RDBMS's.

One can substitute other web servers for Apache. Normally LedgerSMB is run as a CGI program but it may be possible to run it in the web server process (note that this may not be entirely thread-safe).

The operating system can be any that supports a web server and Perl (since PostgreSQL need not run on the same system). However, there are a few issues running LedgerSMB on Windows (most notably in trying to get Postscript documents to print properly).

On the client side, any web-browser will work. Currently, the layout is different for Lynx (which doesn't support frames), and the layout is not really useful under eLinks (the replacement for Lynx which does support frames). Some functionality requires Javascript to work properly, though the application is usable without these features.

## 21.2 Capacity Planning

Some companies may ask how scalable LedgerSMB is. In general, it is assumed that few companies are going to have a need for a high-concurrency accounting system. However, with all the features available in LedgerSMB, the staff that may have access to some of the application may be senior enough to make the question worthwhile.

As of 1.3, there are users with databases containing over a million financial transactions, thousands of customers and vendors, and a full bookkeeping department. In general, 1.3 is considered scalable for midsize businesses.

In general if you are going to have a large number of users for the software, or large databases, we'd generally suggest running the database on a separate server, since this makes it easier to isolate and address performance issues. Additionally, the database server is the hardest part to scale horizontally and so you want to put more hardware there than with the database server.

### 21.2.1 Scalability Strategies

LedgerSMB is a fairly standard web-based application. However, sometimes the database schema changes during upgrades. In these cases, it becomes impossible to use different versions of the software against the same database version safely. LedgerSMB checks the version of the database and if the version is higher than the version of the software that is running, will refuse to run.

Therefore although one strategy might be to run several front-end web servers with LedgerSMB, in reality this can be a bit of a problem. One solution is to take half of the front-end servers off-line while doing the initial upgrade, and then take the other offline to upgrade when these are brought back online.

The database manager is less scalable in the sense that one cannot just add more database servers and expect to carry on as normal. However, aside from the known issues listed below, there are few performance issues with it. If complex reports are necessary, these can be moved to a replica database (perhaps using Slony-I or the streaming repliaction of PostgreSQL 9.x).

If this solution is insufficient for database scalability, one might be able to move staff who do not need real-time access to new entries onto a PG-Pool/Slony-I cluster where new transactions are entered on the master and other data is looked up on the replica. In certain circumstances, one can also offload a number of other queries from the master database in order to minimize the load. LedgerSMB has very few issues in the scalability of the application, and those we find, we correct as quickly as possible.

### 21.2.2 Database Maintenance

PostgreSQL uses a technique called Multi-version Concurrency Control (MVCC) to provide a snapshot of the database at the beginning of a statement or transaction (depending on the transaction isolation level). When a row is updated, PostgreSQL leaves the old row in the database, and inserts a new version of that row into the table. Over time, unless those old rows are removed, performance can degrade as PostgreSQL has to search through all the old versions of the row in order to determine which one ought to be the current one.

Due to the way the SQL statements are executed in LedgerSMB, many inserts will also create a dead row.

A second problem occurs in that each transaction is given a transaction id. These id's are numbered using 32-bit integers. If the transaction id wraps around (prior to 8.1), data from transactions that appear (due to the wraparound) to be in the future suddenly becomes inaccessible. This problem was corrected in PostgreSQL 8.1, where the database will refuse to accept new transactions if the transaction ID gets too close to a wraparound. So while the problem is not as serious in 8.1, the application merely becomes inaccessible rather than displaying apparent data loss. Wraparound would occur after about a billion transactions between all databases running on that instance of PostgreSQL.

Prior to 8.1, the main way to prevent both these problems was to run a periodic vacuumdb command from cron (UNIX/Linux) or the task scheduler (Windows). In 8.1 or later, autovacuum capabilities are part of the back-end and can be configured with the database manager. See the PostgreSQL documentation for treatment of these subjects.

In general, if performance appears to be slowly degrading, one should try to run vacuumdb -z from the shell in order to attempt to reclaim space and provide the planner with accurate information about the size and composition of the tables. If this fails, then one can go to other methods of determining the bottleneck and what to do about it.

#### 21.2.3 Known issues

There are no known issues in PostgreSQL performance as relates to LedgerSMB performance in supported versions.

# 22 Customization Possibilities

LedgerSMB is designed to be customized relatively easily and rapidly. In general, the source code is well written and compartmentalized. This section covers the basic possibilities involving customization.

## 22.1 Brief Guide to the Source Code

LedgerSMB is an application with over 120000 lines of code.[5] While it is not possible to cover the entire application here, a brief overview of the source code is in order.

In the root of the install directory, one will find a setup.pl program, a number of other .pl programs, and a number of directories. The setup.pl program is used to update or install LedgerSMB. The other .pl programs provide a basic set of services for the framework (including authentication) and then pass the work on to the data entry screen file in the bin directory.

The bin directory contains the workflow scripts inherited from SQL-Ledger. These do basic calculations and generate the user interface. The scripts/ directory does the same with new code.

The css directory in the root install directory contains CSS documents to provide various stylesheets one can select for changing various aspects of the look and feel of the application.

The locale directory contains translation files that LedgerSMB uses to translate between different languages. One could add translations to these files if necessary.

The UI directory contains user interface templates for old and new code. These templates use TemplateToolkit to generate the basic HTML. The directory also contains template-specific javascript and css files.

The LedgerSMB directory is where the Perl modules reside that provide the core business logic in LedgerSMB. These modules provide functionality such as form handling, email capabilities, and access to the database through its at least partially object oriented API.

[5] The line count provided by Ohloh.net is 216000 lines of code, but this includes database documentation, the source for this manual, and more. The source code is certainly above 120000 lines by any metric though.

Finally, the sql directory provides the database schemas and upgrade scripts.

## 22.2 Data Entry Screens

One can customize the data entry screens to optimize work flow, display additional information, etc.

### 22.2.1 Examples

We set up hot keys for payment lines, automatically focused the keyboard on the last partnumber field, removed separate print and post buttons to ensure that invoices were always printed and posted together, and removed the ability to print to the screen, and even the ability to scan items in when an invoice was received (using a portable data terminal) and import this data into LedgerSMB. Finally we added the ability to reconcile the till online in a paperless manner.

For another customer, we added the ability to print AR invoices in plain text format and added templates (based on the POS sales template) to do this.

## 22.3 Extensions

One can add functionality to the Perl modules in the LSMB directory and often add missing functions easily.

### 22.3.1 Examples

For one customer, we added a module to take data from a portable data terminal collected when inventory items were taken and use that to add shrinkage and loss adjustments. We also extended the parts model to add a check id flag (for alcohol sales) and added this flag to the user interface.

For another customer, we added a complex invoice/packing slip tracking system that allowed one to track all the printed documents associated with an order or invoice.

## 22.4 Templates

As noted before templates can be modified or extended, though sometimes this involves extending the user interface scripts. Most templates are easy enough to modify.

### 22.4.1 Examples

For one customer we added text-only invoices for AR and AP transactions/Invoices and an ability to use Javascript in them to automatically print them on load.

## 22.5 Reports

The fact that all the data is available within the database manager is a huge advantage of LedgerSMB over Quickbooks and the like. The rapid development of reports allows for one to easily develop reports of any sort within LedgerSMB.

Beginning in 1.4, LedgerSMB has a reporting framework that allows you to turn general SQL queries into reports that can be output in a variety of formats including HTML, PDF, CSV, and ODS. This reporting framework is not too difficult to learn. This section guides you through how to write such a report in LedgerSMB 1.4.

In this example we will walk through the implementation of the Contact Search structure. This is believed to be a fairly basic report. Many features will be mentioned in passing along with which files they are used in, so if you need to use advanced features you will know where to look (I recommend checking the development documentation in the POD of the actual program modules as your primary resource here).

### 22.5.1 Essential Parts of a Report

A report in LedgerSMB 1.4 consists of the following basic components:

- Filter screen
- Stored Procedure
- Report module
- Workflow hooks

Each of these will be covered in turn.

### 22.5.2 Filter Screen

Every filter screen must have a unique name and is placed in UI/Reports/filters/contact_search.html. This file creates an HTML form that submits the search criteria back to a specified script. It is involed through a URL like http://myhost/ledgersmb/reports.pl?action=being_report\&report_name=contact_search\&module=gl. The arguments passed by the HTTP request are: action (what to do), report_name (which filter file to render), and module (which set of business reporting units we might be interested in). Most commonly the correct module is gl which is for all financial transactions.

As always the filter screen is an HTML template rendered with Template Toolkit, but using SGML process instruction tags rather than the default %-based ones. The choice of the SGML <?lsmb ... ?> PI tags is largely to support LaTeX better as well as to better conform to HTML and SGML standards.

You may also want to look at UI/lib/report_base.html for prebuilt controls for things like date ranges and lists of business report units.

### 22.5.3 The Stored Procedure

The primary work for the report is the stored procedure. There are a couple general LedgerSMB rules to be aware of. Prefix your arguments with in_ and it's generally better to let the query mapper generate the queries than try to pass args in directly. Typically package and method name are separated by a double underscore (__) which can be a bit hard to see sometimes.

The stored procedure for the Contact Search report is defined as follows:

```
CREATE OR REPLACE FUNCTION contact__search
(in_entity_class int, in_contact text, in_contact_info text[],
        in_meta_number text, in_address text, in_city text, in_state text,
        in_mail_code text, in_country text, in_active_date_from date,
        in_active_date_to date,
        in_business_id int, in_name_part text, in_control_code text,
        in_notes text)
RETURNS SETOF contact_search_result AS $$
DECLARE
        out_row contact_search_result;
        loop_count int;
        t_contact_info text[];
BEGIN
        t_contact_info = in_contact_info;


        FOR out_row IN
                SELECT e.id, e.control_code, ec.id, ec.meta_number,
```

```
                ec.description, ec.entity_class,
                c.legal_name, c.sic_code, b.description , ec.curr::text
        FROM (select * from entity
               where control_code like in_control_code || '%'
              union
              select * from entity where in_control_code is null) e
        JOIN (SELECT legal_name, sic_code, entity_id
                FROM company
               WHERE legal_name @@ plainto_tsquery(in_name_part)
              UNION ALL
              SELECT legal_name, sic_code, entity_id
                FROM company
               WHERE in_name_part IS NULL
              UNION ALL
             SELECT coalesce(first_name, '') || ' '
                    || coalesce(middle_name, '') || ' '
                    || coalesce(last_name, ''), null, entity_id
               FROM person
              WHERE coalesce(first_name, '') || ' '
                    || coalesce(middle_name, '') || ' '
                    || coalesce(last_name, '')
                     @@ plainto_tsquery(in_name_part)
              UNION ALL
             SELECT coalesce(first_name, '') || ' '
                    || coalesce(middle_name, '') || ' '
                    || coalesce(last_name, ''), null, entity_id
               FROM person
               WHERE in_name_part IS NULL) c ON (e.id = c.entity_id)
        LEFT JOIN entity_credit_account ec ON (ec.entity_id = e.id)
        LEFT JOIN business b ON (ec.business_id = b.id)
        WHERE coalesce(ec.entity_class,e.entity_class) = in_entity_class
                AND (c.entity_id IN (select entity_id
                                       FROM entity_to_contact
                                      WHERE contact ILIKE
                                                  ANY(t_contact_info))
                                            OR '' ILIKE
                                                  ALL(t_contact_info)
                                            OR t_contact_info IS NULL)

                AND ((in_address IS NULL AND in_city IS NULL
                                 AND in_state IS NULL
                                 AND in_country IS NULL)
                        OR (c.entity_id IN
                        (select entity_id FROM entity_to_location
                        WHERE location_id IN
                                (SELECT id FROM location
                                WHERE (line_one @@ plainto_tsquery(
in_address)
                                       OR
                                       line_two @@ plainto_tsquery(
                                                   in_address)
                                       OR
```

```
                                                line_three @@ plainto_tsquery(
                                                             in_address))
                                                 AND city ILIKE
                                                        '%' ||
                                                        coalesce(in_city, '')
                                                        || '%'
                                                 AND state ILIKE
                                                        '%' ||
                                                        coalesce(in_state, '')
                                                        || '%'
                                                 AND mail_code ILIKE
                                                        '%' ||
                                                        coalesce(in_mail_code,
                                                                 '')
                                                        || '%'
                                                 AND country_id IN
                                                        (SELECT id FROM country
                                                        WHERE name ilike
                                                              in_country
                                                                OR short_name
                                                                ilike
                                                                in_country)))))
                        AND (ec.business_id =
                            coalesce(in_business_id, ec.business_id)
                            OR (ec.business_id IS NULL
                                    AND in_business_id IS NULL))
                        AND (ec.startdate <= coalesce(in_active_date_to,
                                        ec.startdate)
                            OR (ec.startdate IS NULL))
                        AND (ec.enddate >= coalesce(in_active_date_from,
ec.enddate)
                            OR (ec.enddate IS NULL))
                        AND (ec.meta_number like in_meta_number || '%'
                             OR in_meta_number IS NULL)
                        AND (in_notes IS NULL OR e.id in (
                                    SELECT entity_id from entity_note
                                     WHERE note @@ plainto_tsquery(in_notes))
                                  OR ec.id IN (select ref_key FROM eca_note
                                     WHERE note @@ plainto_tsquery(in_notes)))
        LOOP
                RETURN NEXT out_row;
        END LOOP;
END;
$$ language plpgsql;
```

This is just a bit complex. Most of the query is just a set of filter condutions, however. The filter runs, and spits out results based on inputs. Moreover, because of the magic of the DBObject mapper, an input in the filter screen named notes gets mapped to the in_notes argument here.

### 22.5.4 Report Module

The report module for this report is in LedgerSMB/DBObject/Report/Contact/Search.pm. This file defines the basic outlay of the report. These modules all must inherit from LedgerSMB::DBObject::Report and therefore must also use Moose. This also automatically makes the DBOBject query mapping routines available to report objects.

All reports are assumed to be tabular structures (this isn't always the case but non-tabular reports are an advanced topic beyond this current introduction. See the income statement and balance sheet reports for how this would be done however).

All criteria are handled as Moose properties (using the "has" function) and it is important that they be listed here. Non-listed properties will not be available to your stored procedure. Reports are also assumed to have rows and so the Report class automatically handles this here.

Additionally the following functions are required in your module:

**columns** This returns a list of column data. Each is a hashref with the following attributes:

> **col_id** The unique name of the column
>
> **type** Typically 'text' though 'checkbox' and 'href' are not uncommon.
>
> **href_base** Only valid for type href, and is the beginning of the href.
>
> **pwidth** Size to print on an arbitrary scale, for PDF only.
>
> **name** Label for the column

**name** This returns the name of the report, to print in the header

**header_lines** Returns a list of hasrefs (name/text are keys) for printing criteria information on the header of the report.

Additionally subtotal_on (returning a list of column names for subtotals) is optional.

Typically the convention is that a prepare_input function takes a $request object and turns dates into appropriate types, while a run_report function actually runs the report and sets the rows property. render($request) is provided by the Report class (see workflow hooks below).

### 22.5.5 Workflow Hooks

In the workflow script one will typically see some lines like:

```
sub search{
    my ($request) = @_;

    LedgerSMB::DBObject::Report::Contact::Search->prepare_criteria($request);
    my $report = LedgerSMB::DBObject::Report::Contact::Search->new(%$request);
    $report->run_report;
    $report->render($request);
}
```

There are a few notes about the above lines that are worth making though usually these can be copied and pasted, and modified to fit.

The render call requires a request object because it checks to see if a set of columns are requested. If any columns are explicitly requested it displays only those that were requested. If columns are not explicitly requested it displays all available columns. A column is explicitly requested if the request contains a true value for a parameter of col_$colname, so a column named city would be requested by a variable set named col_city. Additionally ordering and subtotals are then automatically handled as well.

# 23 Integration Possibilities

An open database system and programming API allows for many types of integration. There are some challenges, but in the end, one can integrate a large number of tools.

## 23.1 Reporting Tools

Any reporting tool which can access the PostgreSQL database can be used with LedgerSMB for custom reporting. These can include programs like Microsoft Access and Excel (using the ODBC drivers), PgAccess (A PostgreSQL front-end written in TCL/Tk with a similar feel to Access), Rekall, Crystal Reports, OpenOffice and more.

### 23.1.1 Examples

We have created spreadsheets of the summaries of activity by day and used the ODBC driver to import these into Excel. Excel can also read HTML tables, so one can use PostgreSQL to create an HTML table of the result and save it with a .xls extension so that Windows opens it with Excel. These could then be served via the same web server that serves LedgerSMB.

## 23.2 Line of Business Tools on PostgreSQL

Various line of business tools have been written using PostgreSQL in large part due to the fact that it is far more mature than MySQL in areas relating to data integrity enforcement, transactional processing, and the like. These tools can be integrated with LedgerSMB in various ways. One could integrate this program with the HERMES CRM framework, for example.

### 23.2.1 Strategies

In general, it is advisable to run all such programs that benefit from integration in the same database but under different schemas. This allows PostgreSQL to become the main method of synchronizing the data in real time. However, sometimes this can require dumping the database, recreating the tables etc. in a different schema and importing the data back into LedgerSMB.

One possibility for this sort of integration is to use database triggers to replicate the data between the applications in real-time. This can avoid the main issue of duplicate id's. One issue that can occur however relates to updates. If one updates a customer record in HERMES, for example, how do we know which record to update in LedgerSMB? There are solutions to this problem but they do require some forethought.

A second possibility is to use views to allow one application to present the data from the other as its own. This can be cleaner regarding update issues, but it can also pose issues regarding duplicate id fields.

### 23.2.2 Examples

Others have integrated L'ane POS and LedgerSMB in order to make it work better with touch screen devices. Still others have successfully integrated LedgerSMB and Interchange. In both cases, I believe that triggers were used to perform the actual integration.

## 23.3 Line of Business Tools on other RDBMS's

Often there are requests to integrate LedgerSMB with applications like SugarCRM, OSCommerce, and other applications running on MySQL or other database managers. This is a far more complex field and it requires a great deal more effort than integrating applications within the same database.

### 23.3.1 Strategies

Real-time integration is not always possible. MySQL does not support the SQL extension SQL/MED (Management of External Data) that supports non-MySQL data sources so it is not possible to replicate the data in real-time. Therefore one generally resorts to integrating the system using time-based updates. Replication may be somewhat error-prone unless the database manager supports triggers (first added to MySQL in 5.0) or other mechanisms to ensure that all changed records can be detected and replicated. In general, it is usually advisable to add two fields to the record– one that shows the insert time and one that shows the last update.

Additionally, I would suggest adding additional information to the LedgerSMB tables so that you can track the source record from the other application in the case of an update.

In general, one must write replication scripts that dump the information from one and add it to the other. This must go both ways.

### 23.3.2 Integration Products and Open Source Projects

While many people write Perl scripts to accomplish the replication, an open source project exists called DBI-Link. This package requires PL/Perl to be installed in PostgreSQL, and it allows PostgreSQL to present any data accessible via Perl's DBI framework as PostgreSQL tables. DBI-Link can be used to allow PostgreSQL to pull the data from MySQL or other database managers.

DBI-Link can simplify the replication process by reducing the operation to a set of SQL queries.

# 24 Customization Guide

This section is meant to provide a programmer with an understanding of the technologies enough information to get up to speed quickly and minimize the time spent familiarizing themselves with the software. Topics in this section are listed in order of complexity. As it appeals to a narrower audience than previous discussions of this topic, it is listed separately.

## 24.1 General Information

The main framework scripts (the ar.pl, ap.pl, etc. scripts found in the root of the installation directory) handle such basic features as instantiating the form object, ensuring that the user is logged in, and the like. They then pass the execution off to the user interface script (usually in the bin/mozilla directory).

LedgerSMB in many ways may look sort of object oriented in its design, but in reality, it is far more data-driven than object oriented. The Form object is used largely as a global symbol table and also as a collection of fundamental routines for things like database access. It also breaks down the query string into sets of variables which are stored in its attribute hash table.

In essence one can and often will store all sorts of data structures in the primary Form object. These can include almost anything. It is not uncommon to see lists of hashes stored as attributes to a Form object.

## 24.2 Customizing Templates

Templates are used to generate printed checks, invoices, receipts, and more in LedgerSMB. Often the format of these items does not fit a specific set of requirements and needs to be changed. This document will not include LaTeX or HTML instruction, but will include a general introduction to editing templates. Also, this is not intended to function as a complete reference.

Template instructions are contained in tags <?lsmb and ?>. The actual parsing is done by the parse_template function in LSMB/Form.pm.

### 24.2.1 Template Control Structures

As of 1.3, all templates use the Template Toolkit syntax for generating LaTeX, text, and html output. The LaTeX can then be processed to create postscript or pdf files, and could be trivially extended to allow for DVI output as well.

Template Toolkit provides a rich set of structures for controlling flow which are well beyond what was available in previous versions of LedgerSMB. The only difference is in the start and end tag sequences, where we use <?lsmb and ?> in order to avoid problems with rendering LaTeX templates for testing purposes.

## 24.3 Customizing Forms

Data entry forms and other user interface pieces are in the bin directory. In LedgerSMB 1.0.0 and later, symlinks are not generally used.

Each module is identified with a two letter combination: ar, ap, cp, etc. These combinations are generally explained in the comment headers on each file.

Execution in these files begins with the function designated by the form->{action} variable. This variable is usually derived from configuration parameters in the menu.ini or the name of the button that was clicked on to submit the previous page. Due to localization requirements, the following process is used to determine the appropriate action taken:

The $locale->getsub routine is called. This routine checks the locale package to determine if the value needs to be translated back into an appropriate LSMB function. If not, the variable is lower-cased, and all spaces are converted into underscores.

In general there is no substitute for reading the code to understand how this can be customized and how one might go about doing this.

In 1.3, all UI templates for new code (referenced in the scripts directory) and even some for the old code (from the bin directory) use TemplateToolkit and follow the same rules as the other templates.

## 24.4 Customizing Modules

The Perl Modules (.pm files) in the LedgerSMB directory contain the main business logic of the application including all database access. Most of the modules are relatively easy to follow, but the code quality leaves something to be desired. The API calls themselves are likely to be replaced in the future, so work on documenting API calls is now focused solely on those calls that are considered to be stable. At the moment, the best place to request information on the API's is on the Developmers' Email List (ledger-smb-devel@lists.sourceforge.net).

Many of these modules have a fair bit of dormant code in them which was written for forthcoming features, such as payroll and bills of materials.

One can add a new module through the normal means and connect it to other existing modules.

### 24.4.1 Database Access

Both the Form (old code) and LedgerSMB (new code) objects have a dbh property which is a database handle to the PostgreSQL database. This handle does not autocommit.

# Part III
# Appendix

## A Where to Go for More Information

There are a couple of relevant sources of information on LedgerSMB in particular.

The most important resources are the LedgerSMB web site (http://www.ledgersmb.org) and the email lists found at our Sourceforge site.

In addition, it is generally recommended that the main bookkeeper of a company using LedgerSMB work through at least one accounting textbook. Which textbook is not as important as the fact that a textbook is used.

## B Quick Tips

### B.1 Understanding Shipping Addresses and Carriers

Each customer can have a default shipping address. This address is displayed prominantly in the add new customer screen. To change the shipping address for a single order, one can use the ship to button at the bottom of the quote, order, or invoice screen.

The carrier can be noted in the Ship Via field. However, this is a freeform field and is largely used as commentary (or instructions for the shipping crew).

### B.2 Handling bad debts

In the event that a customer's check bounces or a collection requirement is made, one can flag the customer's account by setting the credit limit to a negative number.

If a debt needs to be written off, one can either use the allowance method (by writing it against the contra asset account of "Allowance for Bad Debts" or using the direct writeoff method where it is posted as an expense.

## C Step by Steps for Vertical Markets

### C.1 Common Installation Errors

- LedgerSMB is generally best installed in its own directory outside of the wwwroot directory. While it is possible to install it inside the wwwroot directory, the instructions and the faq don't cover the common problems here.
- When the chart of accounts (COA) is altered such that it is no longer set up with appropriate items, you can make it impossible to define goods and services properly. In general, until you are familiar with the software, it is best to rename and add accounts rather than deleting them.

### C.2 Retail With Light Manufacturing

For purposes of this example we will use a business that assembles computers and sells them on a retail store.

1. Install LedgerSMB.
2. Set preferences, and customize chart of accounts.

(a) Before customizing the COA it is suggested that you consult an accountant.

3. Define Goods, Labor, and Services as raw parts ordered from the vendors.
   - These are located under the goods and services menu node.
4. Define assemblies.
   - These are also located under goods and services.
   - Component goods and services must be defined prior to creating assembly.
5. Enter an AP Invoice to populate inventory with proper raw materials.
   - One must generally add a generic vendor first. The vendor is added under AP->Vendors->Add Vendor.
6. To pay an AP invoice like a check, go to cash->payment. Fill out approrpiate fields and click print.
   - Note that one should select an invoice and enter in the payment amount in the appropriate line of the invoice list. If you add amounts to the master amount list, you will find that they are added to the amount paid on the invoice as a prepayment.
   - The source field is the check number.
7. Stock assemblies
8. One can use AR Invoices or the POS interface to sell goods and services.
   - Sales Invoice
     - Can be generated from orders or quotations
     - Cannot include labor/overhead except as part of an assembly
     - One can make the mistake of printing the invoice and forgetting to post it. In this event, the invoice does not officially exist in the accounting system.
     - For new customers, you must add the customer first (under AR->Customers->Add Customer.
   - POS Interface
     - Cannot include labor/overhead except as part of an assembly
     - Printing without posting is often even easier in the POS because of the rapid workflow. Yet it is just as severe a problem.
   - Ecommerce and Mail Order Operations
     - See the shipping workflow documentation above.
   - Customers are set up by going to AR->Customers->Add Customer (or the equivalent localized translation). The appropriate fields are filled out and the buttons are used at the bottom to save the record and optionally use it to create an invoice, etc.
     - Saving a customer returns to the customer screen. After the appropriate invoice, transaction, etc. is entered and posted, LedgerSMB will return to the add customer screen.
9. One can use the requirements report to help determine what parts need to be ordered though one cannot generate PO's directly from this report.

Note, the needs of LedgerSMB are mostly useful for light manufacturing operations (assembling computers, for example). More manufacturing capabilities are expected to be released in the next version.

A custom assembly is a bit difficult to make. One must add the assembly prior to invoice (this is not true of goods and services). If the assembly is based on a different assembly but may cost more (due to non-standard parts) you can load the old assembly using Goods and Services->Reports->Assemblies and then make necessary changes (including to the SKU/Partnumber) and save it as new.

Then one can add it to the invoice.

# D Glossary

**BIC** Bank Identifier Code is often the same as the S.W.I.F.T. code. This is a code for the bank a customer uses for automated money transfers.

**COGS** is Cost of Goods Sold. When an item is sold, then the expense of its purchase is accrued as attached to the income of the sale. It is tracked as COGS.

**Credit** : A logical transactional unit in double entry accounting. It is the opposite of a debit. Credits affect different account types as follows:

**Equity** : Credits are added to the account when money is invested in the business.

**Asset** : Credits are added when money is deducted from an asset account.

**Liability** : Credits are added when money is owed to the business account.

**Income** : Credits are added when income is earned.

**Expense** : Credits are used to apply adjustments at the end of accounting periods to indicate that not all the expense for an AP transaction has been fully accrued.

**Debit** : A logical transactional unit in double entry accounting systems. It is the opposite of a credit. Debits affect different account types as follows:

**Equity** : Debits are added when money is paid to business owners.

**Asset** : Debits are added when money is added to an account.

**Liability** : Debits are added when money that is owed is paid off.

**Income** : Debits are used to temporarily adjust income to defer unearned income to the next accounting period.

**Expense** : Debits are added as expenses are incurred.

**IBAN** International Bank Account Number is related to the BIC and is used for cross-border automated money transfers.

**List** Price is the recommended retail price.

**Markup** is the percentage increase that is applied to the last cost to get the sell price.

**ROP** Re-order point. Items with fewer in stock than this will show up on short reports.

**Sell** Price is the price at which the item is sold.

**SKU** Stock Keeping Unit: a number designating a specific product.

**Source** Document : a paper document that can be used as evidence that a transaction occurred. Source documents can include canceled checks, receipts, credit card statements and the like.

**Terms** is the number of days one has to pay the invoice. Most businesses abbreviate the terms as Net n where n is the number of days. For example, Net 30 means the customer has 30 days to pay the net due on an invoice before it is late and incurs late fees. Sometimes you will see 2 10 net 30 or similar. This means 2% discount if paid within 10 days but due within 30 days in any case.