The Cclite Manual

Table of Contents

Introduction

Cclite is software, written in Perl, for community currencies and local exchange trading systems (LETS). It allows multiple currencies within one trading group and exchange between different trading groups. It has multiple interfaces and (recently) an osCommerce™ (http://www.oscommerce.com/) gateway. In this version, there is also a much improved SMS gateway and rough software for Tiki-Wiki, Drupal, Joomla and Elgg passthrough. This manual has (at least) three distinct purposes:

- Manual for potential and actual users of Cclite
- Explanation of philosophy, design decisions and directions
- Some use cases for Cclite

The current shape of the software probably reflects my view of alternative currencies. It isn't the only one and I've included a web resources chapter at the end, for those who want to read about origins, theory and controversies.

Thanks to Mary Fee (of Letslink Uk), Michael Linton (of LETS and openmoney), Saul Albert and Jo Walsh and all others with whom I've been discussing this sporadically. Thanks to Sook-Yin Chow in whose appartment I settled down to start writing this in about 2004, ten years already.

Restaurant Guide

I have also been in the Da Lat in Belleville (opposite the metro) and 149 Avenue de Choisy eating pho bo tai, the MSG works wonders too.

Now back in London, I've been eating next door to the Geffrye Museum at the Loong Kee Cafe, nice fresh lemon drink here too, also. I'd also recommend Govinda's run by the Hare Krishna temple in Soho Street and 'Indian Veggie' at the far end of Chapel Market in Islington [now £6 much better food, hipster problem developing].

New World in Gerrard Place is the nearest thing to Hong Kong, outside of the place itself, nice Lo Bak Go, crispy! Shanghai in Dalston is pretty good too, but beware huge numbers of hipsters, poke and slap at them, if necessary.

For meat, then the Lahore Kebab in Umberston Street is pretty famous. Tayyabs in Fieldgate Street is pretty good too, great chicken tikka, mind the queues in the evening, and the Curry Hut in Chrisp Street market is 'robust'. The Anchor Cafe in Salmon Lane is a good fry-up too. I'll add the Taste of India in East Ham, great food and welcome.

Getting Help with Cclite

If you want to help with Cclite use Googlegroups:
`http://groups.google.com/group/cclite` you may need to be patient. For other questions use the contact form at `http://www.hughbarnard.org`

Conventions Used

Words that are in `courier` are technical, names of scripts, modules, files, urls etc.
Words in *italics* are example names and values.
Word in <words> angles are values to be filled.

In general, command line, program fragments etc. in the text have a shadowed border.

Words and phrases in **Bold** are usually names of menu tab and entries, for example: **Create Currency**

Shaded stuff is usually where some detail has changed since the previous manual.

We don't want any of that dirty BNF stuff (`http://en.wikipedia.org/wiki/Backus-Naur_form`) in this manual!

Buying this manual as a dead tree will support this work and it won't be popular enough to cause any significant deforestation either.

1. Installation

Environment and Installation Overview

The first part of this section deals with upgrading, and the system and software environment for Cclite. You need to read this to se whether you can install it within on the server or within the hosting account that you have chosen. For installation itself, there are three parts:

- Unpacking and Installation
- Configuration
- Setting up a Registry (a trading group, you need at least one)
- Setting up a currency within a registry

The unpacking and installation is 'easier' now for Debian and Ubuntu in that there is a Debian style package for the software. There's also a package laid out for Cpanel.

The software now has a web configuration tool, which makes some of the configuration easier and more accessible outside the command line. There is also a configuration checker which will give a report on whether the necessary modules and programs are on the server. Within a registry (that's a single Mysql database) the registry manager must log on and set up (at least) one valid currency.

Upgrade from Previous Version

These are the main steps. If you have modified templates etc., you may need to save and restore them. Since the user table has been expanded, for example, it needs to use the 'new' templates:

1. Save existing registries before step 2!
2. Apply the contents of all `registry_upgrade<x>.sql,` `registry_upgrade1.sql` `registry_upgrade2.sql` etc. (if you haven't upgraded for a while) to any existing registries.
3. In general, registry upgrades should be applied in order. They are (usually) non-destructive and add additional fields and tables to the database structure. Sometimes they add values to fields with constrained values also. **Please look inside, to see what they do!**
4. Install this version via `tar` or using the Debian package manager, see Unpacking and Installation later in this section.
5. Remove or rename any existing `cclite.cf` (which contains out of date information) and create again using `ccinstall.cgi`

Environment Overview

The current version of Cclite was developed under its own (virtual server) web root on Linux Mint 13. However, there's now a build for Debian, Fedora-style one as a tarball, a provisional RPM package, a Cpanel style one [though there seem to be several configurations of cpanel] and an experimental one for Windows XP.

Cclite can probably be mixed with another application under the same web root but my preference would be subdomaining (which is also a possible solution to community currency namespace and discovery), for example: `cclite.yourdomain.com`.

Requirements

You may need to install Perl modules from CPAN (`http://www.cpan.org`) on your server or ActiveState if you're using Windows. The Apache configuration is a standard configuration and therefore should be OK in a hosted environment. Cclite needs this software:

- A webserver, it’s being developed using Apache 2.0
- You'll need AllowOverride: http://httpd.apache.org/docs/2.0/mod/core.html#allowoverride in the Apache configuration, if you want to use the REST interface in .htaccess
- Perl interpreter, at 5.10.x, the more modern the better!
- MySql at 4.0 or above (InnoDb to allow transactions)
- The Perl Modules in the following table, for extra facilities only
- Gammu and and a mobile phone/dongle, if you are running your own SMS gateway. If you are using an external commercial gateway, SMS messages are processed via `ccsmsgate.cgi` and modules with `lib/Ccsms` These were designed for a specific application/user and gateway and need some modification for your supplier.
- For Windows, there's a potential trouble with mail and Net::SMTP. What works seamlessly on Linux, is problematic on Windows. A workaround for this is hMailServer: http://www.hmailserver.com/ on the same system as Cclite and keeping the Net::SMTP which is default and a core module.

As of version 0.9.x, Log4perl is removed and [if your version of Perl is modern] Cclite should work with core modules only. Some of the extras, phone, GPG, Jabber etc. will not, of course.

Builds and Version Number

There is now a build system, so the various builds have version numbers. These don't mean a whole bunch at the moment, but please quote them when asking questions, this may isolate any problematic builds which will then be removed from sourceforge and updated on github.

Perl Modules for Cclite

From verson 0.9.x Cclite **should install using a modern Perl without any of these extra modules**.

CGI::Carp	This should be installed on most commodity hosters. In the main Cclite scripts, it will send fatals to the browser, rather than giving 500 errors.
Soap::Lite	http://www.soaplite.com only if configured as multiregistry or using SMS collection readsms_from_gammu.pl remotely. This is gradually being replaced by a REST interface.
Digest	
Digest::SHA2	Both Soaplite and Digest need some support packages, if you use perl -MCPAN -e shell or Webmin, they will be suggested at install time.
Bit::Vector	
DBI	You need this for the MySQL interface, it's often supplied with commodity hosting.
File::Path	This should be core, used for cross-platform directory creation
MIME::Base64	**Now needed for tokens etc. This may need re-installing for Redhat.**
MIME::Decoder	Used for GPG Email and Jabber only
GnuPG	Used for GPG Email and Jabber only
NET::Smtp and NET::POP3	New client/server method for mail
Net::XMPP	Used for Jabber only
OpenOffice::OODoc	Used for print_statements.pl and print_yellowpages.pl only
XML::RSS;	If you want to try and use RSS, this is needed for Ccrss.pm:
XML::Simple;	If you want to try and use RSS, this is needed for Ccrss.pm:
LWP::Simple;	I suspect this is dragged in via SOAP anyway!
Net::OpenID::Consumer;	Used by ccopenid.cgi, needed for openid but won't stop anything else working.
Test::More	This should be in the core installation for Perl, only the installation checker uses it.

From version 0.4.0 on, if you configure as `multiregistry=no`, you don't need `SOAP::Lite`. But you may need it for `readsms_from_gammu.pl` if the script and the Cclite installation are on different systems. The easiest way to install the modules is using the Perl CPAN module and the shell command or apt-get install on Debian based systems such as Ubuntu:

```
perl -MCPAN -e shell
>install Soap::Lite
```

Batch processes that produce print documents, such as statements and the initial yellowpages directory use the Open Office modules. So, if you don't print stuff, you don't need these.

Use of Oauth

Cclite is working with others to standardise on Opentransact: http://www.opentransact.org/ as a way of making different currency systems intercommunicate. This is will be done via extension/modification of the REST interface. But Oauth http://search.cpan.org/dist/Net-OAuth/ will probably replace the ad-hoc connection method used by the gateways, currently.

SMS Interface

The local SMS interface also needs `gammu`, if you are going to run your own gateway a Linux utility for handling mobile phones. This is described later. This also calls `lib/Ccsms/Gammu.pm`, so that the gammu method and the commercial gateway method are somewhat unified. This is considerably expanded and tested in 2014, to 'remove' this layer from commercial gateway providers. See later in the manual.

If you use a commercial or other http gateway, you need `ccsmsgateway.cgi` and one of modules in `lib/Ccsms` was written for a specific gateway. This may need some modifications to work with 'your' gateway. There are specific packages for two UK gateway providers (Cardboardfish and AQL) in the package as of summer 2010. If you add a gateway and want to 'give back' it should go into `lib/Ccsms`. Gammu.pm is now more complete, so should be used as a starting point.

MySQL Server

`MySQL` can be installed as an `rpm` or `deb` package on distributions that accept this package format. If you use the Debian package, please install MySql **before installing** Cclite. MySQL is standard on commodity hosting. The next versions may well be MariaDB friendly, for obvious reasons.

REST Interface

If you use the REST interface you need `AllowOverride All` in the Apache configuration for the `public_html` directory., this will let the rewrite rules in `.htaccess` to work. The REST interface will become more significant in the near (2011) future.

Additional Tools

During development, I used `webmin` and `phpmyadmin`. I use `webmin` to set up payment mailboxes and `cron` jobs. This is a matter of taste but a good choice for less command line oriented users. I don't use `phpmyadmin` on test servers because, hopefully) there's no need to change the databases around too much and this can be a major security threat to the databases.

The evolving source code is currently in a `git` repository and there's public access to this at: `https://github.com/hbarnard/cclite` changes made most months/weeks. As of 2008-2010, we are now using `Selenium: http://seleniumhq.org/` to create case based test suites for regression testing. The current regression script is in `/usr/share/cclite/test.`

Unpacking and Install from Tarball

This is a tarball/Fedora style install, follow these steps:

1. Download `cclite-0.9.3.tar.gz` (this is just an example, choose the one that you need from sourceforge) put into a directory in `/tmp` or directly under the new home directory for the domain or subdomain. This is the web root directory not the html directory!
2. `tar –xzvf cclite-0.9.3.tar.gz` to unpack the software
3. Move the directories or the contents of directories (if there's something else within the web root or the web root is not `public_html`) to the web root.
4. Change all the `*.cgi` permissions to world read and world execute and ownership to the webserver, for example:

```
[root@suzette cclite.vectormart.org]# chown -R apache *
[root@suzette cclite.vectormart.org]# chgrp -R apache *
[root@suzette cclite.vectormart.org]# chmod -R a+x cgi-bin
```

Cpanel Installation

The test cpanel hoster has one 'master' account and the additional domains go as public_html under the 'master'. Also `cgi-bin` is underneath public_html that is *WRONG*, but that's cpanel. All the directories that cclite needs are directory under the master home, for example

```
/home/master/lib
/home/master/config
/home/master/public_html/cgi-bin/cclite.cgi
/home/master/public_html/cgi-bin/protected/ccadmin.cgi
```

etc. The rest of this guide depends on that.

NOTE: Since [summer 2010], I've discovered that some cpanel installations, especially those using WHM, expose, for example `/home/admin` as the web root from the file manager. This is much safer, because cgi-bin is above public_html and will probably be OK with either the linux generic or the cpanel package. Because of these variations, you may need some advice, u*se the google group first of all and describe the path to the web root.*

Incidentally, your hoster should allow or undertake Perl module installation, that's the sign of a decent hoster anyway. You can install them locally too, see http://servers.digitaldaze.com/extensions/perl/modules.html for example, a brief guide or search for '*install perl module locally*' on that intertubes web thing.
Here we go:

1. Download `cclite-0.9.3.cpanel.tar.gz` (this is an example) put into the directory in `/home/something`. Unpack and move up all the contents to under `/home/something` if necessary
2. Use `http://yourdomain.org/cgi-bin/protected/checkinstall.cgi` for some initial commentary on whether the install is workable.
3. Use `phpmyadmin` to set up your registry(s) . A registry is just a Mysql database:

   3.1 Create the database, one word only, this is the registry name
   3.2 Use the sql from `sql/registry_0.9.3-sha1.sql` or `sql/registry_0.9.3-sha2.sql` to create the tables depending on whether SHA1 or SHA2 is installed/used. SHA2 is now the default choice.
   3.3 Add a user and password for this database that you'll need in step 4 below, for several registries, the database user will be the same (this will probably change in a future release)
4. Set up mail accounts for the batch processes and for the manager of each registry, you use these in step 7.
5. Use `http://yourdomain.org/cgi-bin/protected/ccinstall.cgi` Change the `dbuser` and `dbpassword` otherwise nothing will work, the rest *may* work as default. If you want all the options, use the **All Options** rather than **Simple** in the drop down at the top.
6. You may need to use cpanelprefix, for example, if your database name is `prefix_prettyname`, this will let you use just `prettyname.` Just put `prefix`, into the **cpanelprefix** field in the install.
7. Now go to `http://yourdomain.org/cgi-bin/cclite.cgi` as user `manager`, password '`manager`'. The registry name is the database name you created in step 3. Then click on the **Admin** link (**Admin Tab** opens it in a tab) .
8. Fill in the registry information, using **Modify** <registry-name> and **Create Currency** (can be more than one) for the registry. The mail accounts are important because, for example the **Manager Email** sends the user confirmation messages.
9. Click on **Create Batch Dirs** if you see directories listed in red on the main page. You may have to create some of these directories manually and change configuration for them, this depends on what your hoster allows.
10. As a matter of security, offline [`chmod a-x`, for example] or remove the installer `ccinstall.cgi` and change the manager's password!

Debian Package Installation

Later in the manual, there's a more complete section on the layout of the Debian package.

Please ***preinstall the*** *mysql-server package* as it's difficult to install it as an automatic dependency package (input root password, for one). Also I had to pre-install libpq5 (`sudo apt-get install libpq5`) on one test system. This server is now in the predepends sections of the Debian control file.

Download the package and double-click on it. If the requirements (Apache [apache2 preferably],. Mysql and various Perl modules) are satisfied, the package will install.

Install scripts should restart Apache as well. If Apache doesn't come up or the site doesn't come up, you may have to use `a2ensite` to enable the site. Some of this, obviously, depends on what your Apache configuration is like, before this package install.

Then go to `http://cclite.private.host` or `http://localhost` where there is an index page with requirements checker (new) and installation links.

The install (except for SMS described later) is as previous releases.

Emergency Package Removal

As this is a preliminary package, it may prove problematic, so you can force its removal with:

```
sudo dpkg --force-all -r cclite
```

The `postrm` script will try and remove configuration files and everything in /usr/share/cclite after standard package removal. This was being left as a 'residue' on the test system therefore, if necessary:

```
sudo dpkg --purge  cclite
```

to remove any remaining configuration files.

RPM Package Installation

This is the first cut of this, as of Spring 2014. So, your mileage may vary, as they say.

Package Install

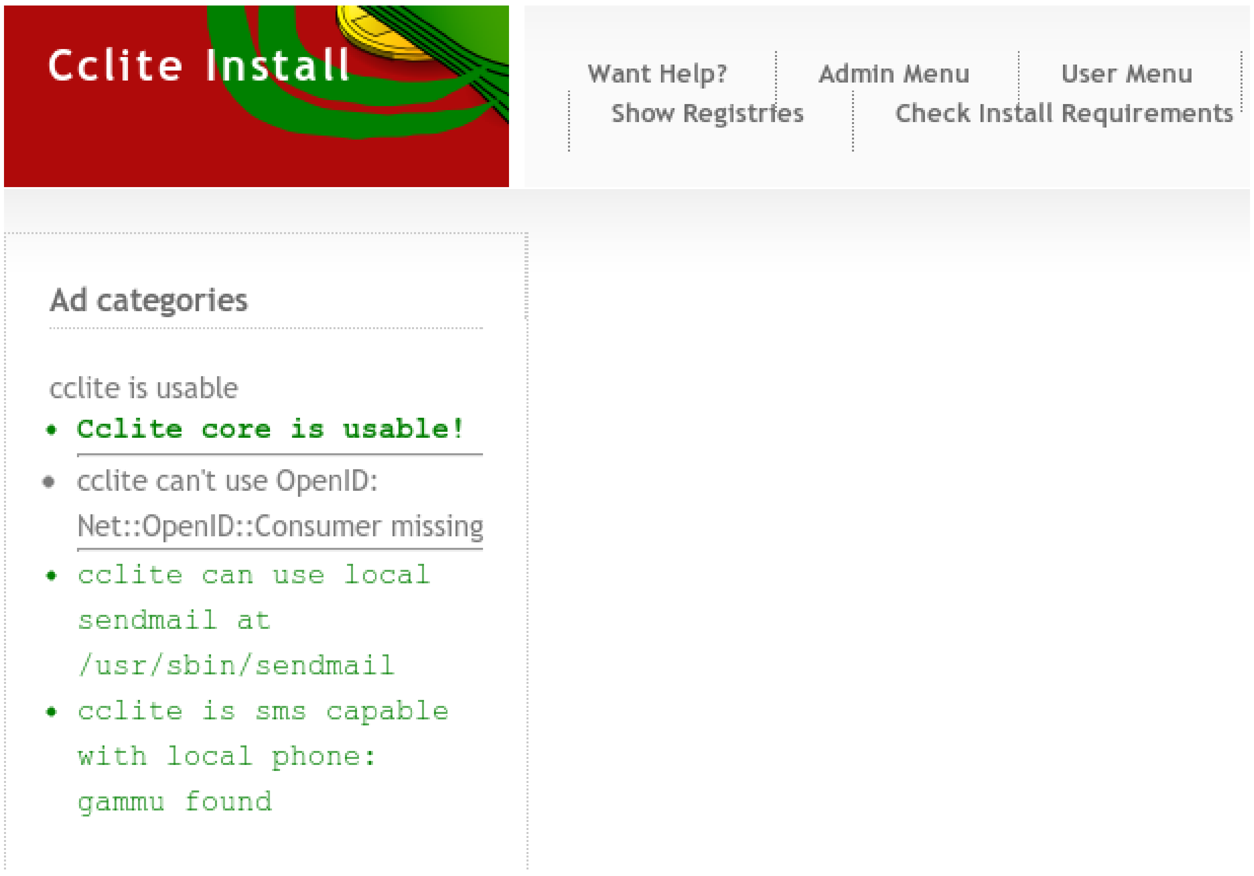
This is the install command:

```
sudo rpm -U -vv -h --nodeps cclite-0.9.3.x86_64.rpm
```
package name my change, depending on the download version

A couple of comments `-vv` is very verbose, lets you see what's happening and `--nodeps` skips dependencies, otherwise *you'll have to have all the extra non-standard Perl modules for openid and SOAP on your system*. You may want these 'anyway'.

Installation Checker

The installation checker is at `http://yourdomain.org/cgi-bin/protected/checkinstall.cgi` and the result will be something like this:



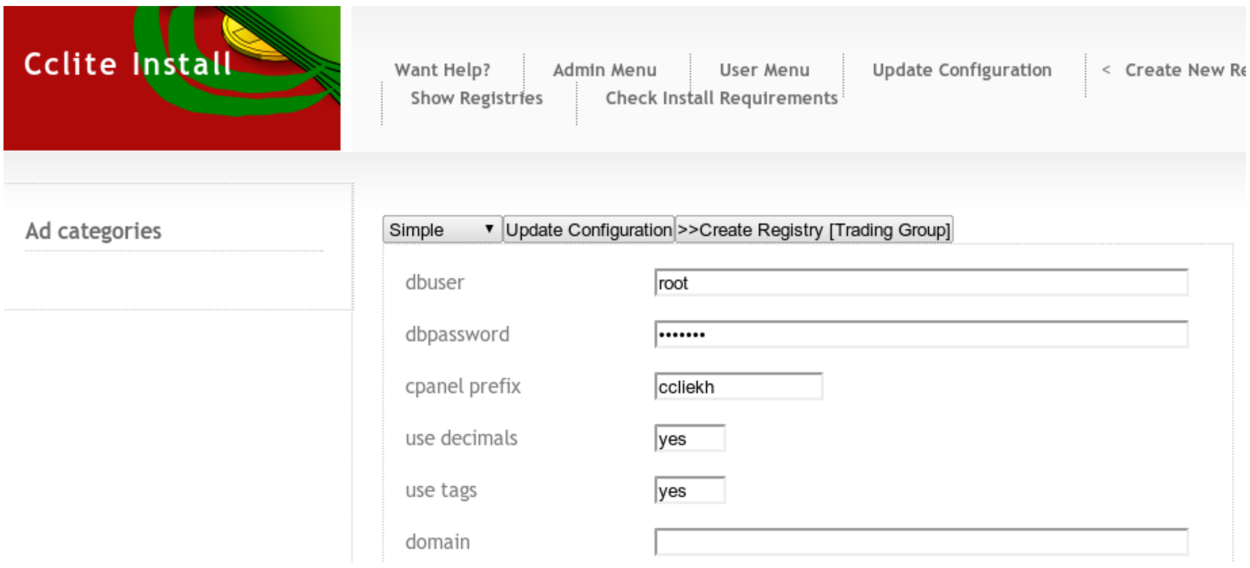
cclite is usable

- **Cclite core is usable!**
- cclite can't use OpenID: Net::OpenID::Consumer missing
- cclite can use local sendmail at /usr/sbin/sendmail
- cclite is sms capable with local phone: gammu found

If there are red entries and cclite is marked as being unusable, the flagged problems need correcting. This usually involves installing extra Perl modules. The check is 'good' but not completely reliable.

Configuration

5. `http://<yourdomain>/cgi-bin/protected/ccinstall.cgi` gives you some default settings for the installation. If you select **All Options** from the drop down, this will unlock all the fields. Many installations should only need database user and password and defaults. Note **use tags** which enables free-form tags on the adverts, this is now preferred.



dbuser	root
dbpassword	•••••••
cpanel prefix	ccliekh
use decimals	yes
use tags	yes
domain	

The `config` subdirectory needs to be writable for this to create or update the configuration file `(cclite.cf)`, otherwise you can cut and paste the resulting configuration with an editor, for example.

Cclite will try and guess most of the parameters based on the domain and the physical path where the software is installed. Here's a key to the values:

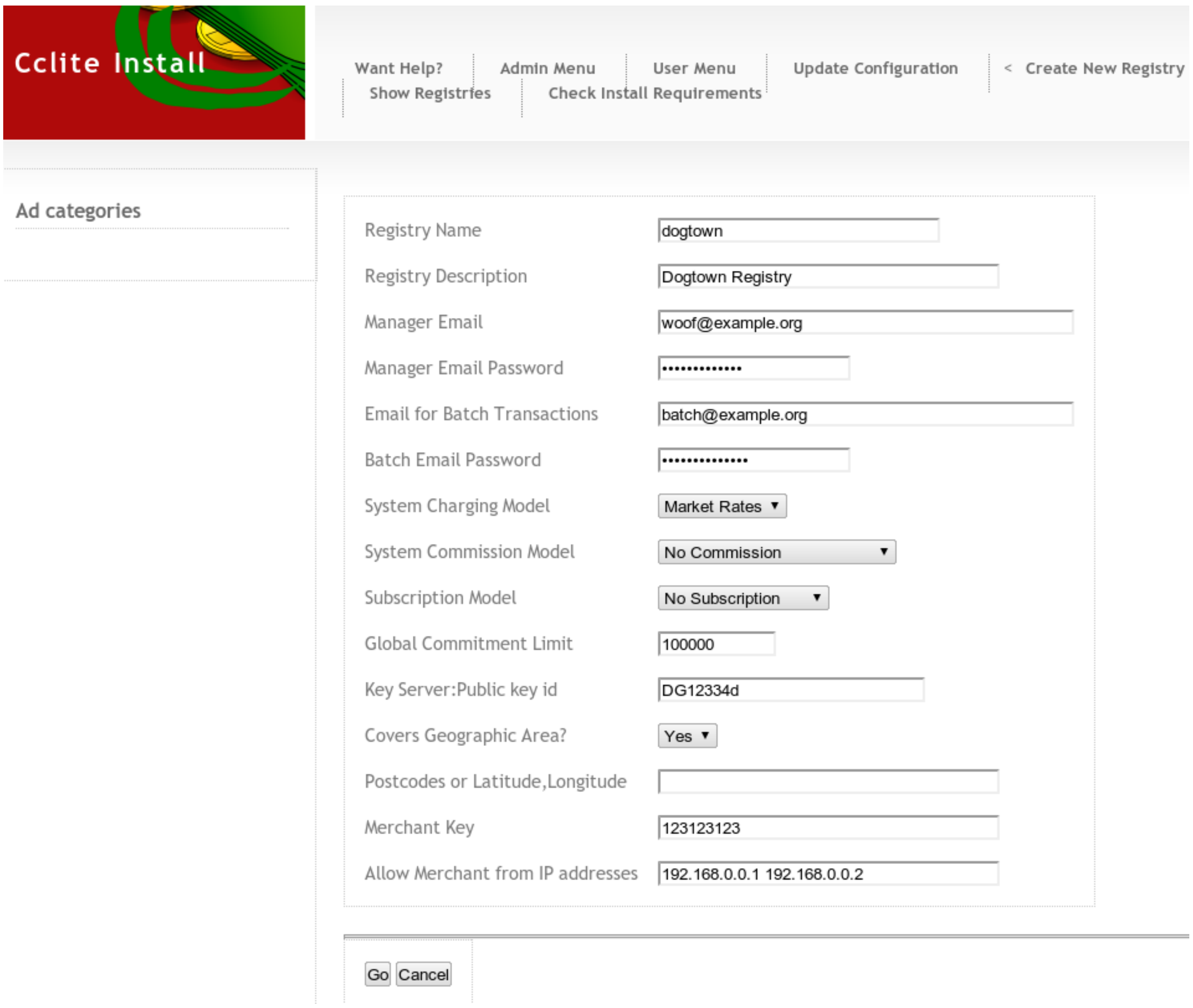
multiregistry	yes/no	SOAP::Lite is used between registries
printdir	/var/www/cclite/public_html/out/print	Directory for print output (future)
linesperpage	15	Screen lines per page for table displays
dbuser	change-me	MySql user
cpanelprefix	<your-prefix-if-necessary>	User prefix if cpanel is used, to make the registry names 'pretty'.
domain	cclite.private.trunk	Domain, should correspond to path
rsspath	/var/www/cclite/public_html/rss	Base path for feed output. Registry name and language are added
systemmailreplyaddress	cclite.noreply@vectormart.org	Return address for system mail
smsout	/var/www/cclite/public_html/out/sms	Output for gammu processed sms
chartdir	/var/www/cclite/public_html/images/charts	Directory for charts, registry added
userss	no	If yes, rss perl modules are needed
home	http://cclite.private.trunk/cgi-bin/cclite.cgi	Main script path
hash_type	sha2	Hash type, which applies to all registries
initialpaymentstatus	waiting	Initial status for a payment can be `waiting` or `accepted`
dbpassword	change-me	Mysql user database password
csvpath	/var/cclite/csv	Input path for uploaded batch files
language	en	Default language. Also still hardcoded
supportmail	support@cclite.3wave.co.uk	Support email, can be used in templates, if necessary
librarypath	/usr/share/cclite/lib	Path to Cclite Perl libraries
net_smtp	1	Use Net::SMTP to send mail, this is preferable as of 0.7.0
version	0.8.1	Current Ccclite Version
csvout	/var/www/cclite/public_html/out/csv	Output path for csv transactions
usedecimals	yes	Use decimals in display. Quantities are assumed to be in cents or pennies when stored [new in 2011]

`usetags`	`yes`	Use free form tags for yellow pages instead of fixed categories.[new in 2011]
`systemmailaddress`	`support@cclite.co.uk`	System email address, mainly the registry based emails are used now
`smspath`	`/var/cclite/sms/inbox`	Inwards path for gammu phone transactions, must correspond with `gammu.cf`
`smslocal`	`1 or 0`	1 [default] if cclite is on the same system as readsms_from_gammu.pl otherwise the SOAP modules are needed
`smsdone`	`/var/www/cclite/public_html/out/sms`	Results of gammu sms processing put here...
`literalspath`	`/usr/share/cclite/literals`	Path for Cclite messages per language
`htmlpath`	`/var/www/cclite/public_html`	Base html path, used mainly for batch process output and charts
`initialuserstatus`	`unconfirmed`	Initial status of user, `active` or `unconfirmed`
`templates`	`/usr/share/cclite/templates/html`	Path to templates

6. Modify any configuration parameters and update. Leave any `notused` entries, as the form checks that everything has a value.

Set up a Registry

7. Use **Create New Registry** entry to create a new registry (a registry is a basic trading group). *If you have already used Cpanel and phpmyadmin to create a database, you don't need to do this*, but you will need to fill in extra information via **Modify <registryname>** in the **Admin** menu. Remember that a registry corresponds to one Mysql database. Here's an example:

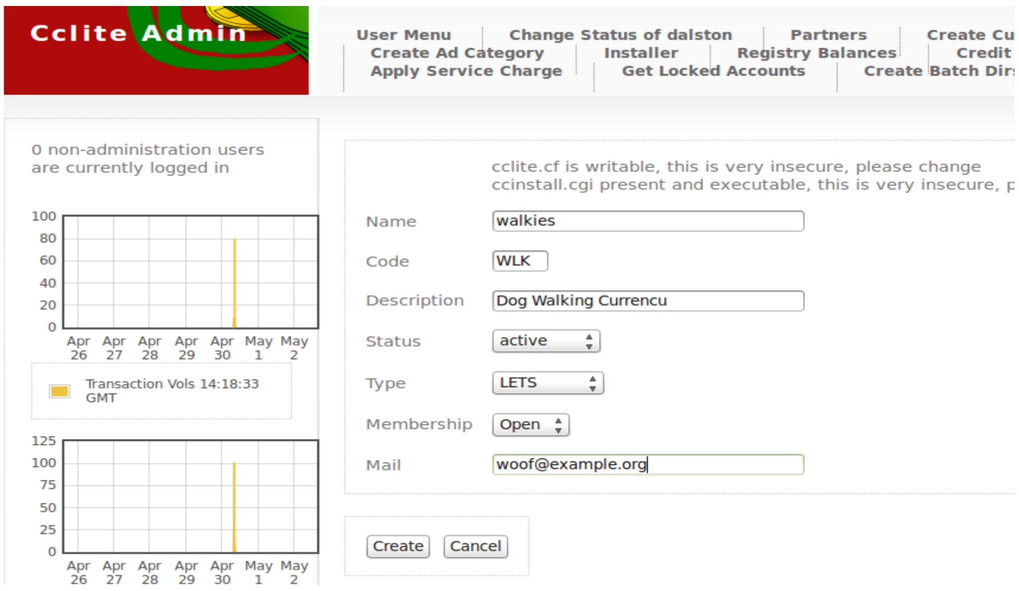


Registry Name	dogtown
Registry Description	Dogtown Registry
Manager Email	woof@example.org
Manager Email Password	•••••••••••••
Email for Batch Transactions	batch@example.org
Batch Email Password	••••••••••••••
System Charging Model	Market Rates
System Commission Model	No Commission
Subscription Model	No Subscription
Global Commitment Limit	100000
Key Server:Public key id	DG12334d
Covers Geographic Area?	Yes
Postcodes or Latitude,Longitude	
Merchant Key	123123123
Allow Merchant from IP addresses	192.168.0.0.1 192.168.0.0.2

Set Up a Currency

8. Go to `http://<yourdomain>/cgi-bin/cclite.cgi` login to the registry as user `'manager'`, password `'manager'` and click on the **Admin** link at the top left.

9. Choose **Create Currency** to create a new currency within your registry, here's an example:



0 non-administration users are currently logged in

cclite.cf is writable, this is very insecure, please change
ccinstall.cgi present and executable, this is very insecure, p

Name	walkies
Code	WLK
Description	Dog Walking Currencu
Status	active
Type	LETS
Membership	Open
Mail	woof@example.org

10. Please change the manager password and off-line `ccinstall.cgi` when you've finished installing. The web interface will always display an error message if this is not done.

11. If using batch processes, make sure that the subdirectory used to write the feeds (as defined by rsspath in the configuration) is writable by the scripts. It needs to belong to the user that runs the web server, for example.

Advanced Install Topics

This describes the optional scripts. Some require additional Perl modules too. They are graphing, mail payment, batch payment, SMS payment (via Gammu) and `rss` offers,wants and matched items.

There is also a script for converting the offers/wants into Open Office format (`print_yellowpages.pl`)and a script for printing statements (`print_statements.pl`). These are not currently in the control panel.

Protecting the out subdirectory in the web root

All results go into subdirectories under the `out` subdirectory in the web root. It may be worth putting an `.htaccess` file into the subdirectories or `out` itself to block access to them all.

Batch Scripts

All these scripts are batch scripts which can now be run from the **Admin** control panel (the opening page when you log into manager). If you want to use them as `cron` scripts, they'll need a little modification (or you could write some Curl!) .

`read_pop_mail_gpg.pl` cannot be run from the control panel currently, probably because the key-ring needs to belong or be accessible by the web user. This may get sorted out, but for the moment it needs to be manual command line [it has an internal and configurable loop in the script] or cron, with a little modification.

These scripts are now all within:
```
<your root>/cgi-bin/protected/batch
<your root>/cgi-bin/protected/graphs
```

If you want to use them as a `cron` job, you need to change the scripts somewhat. These batch components are set up as follows:

- Force the scripts to read a specific configuration by putting one into readconfiguration at the top of each script
- Test batch file on command line, if possible
- Set up `cron` job
- Test complete tool chain

Installing Mail Payment as a Cron

I don't recommend this at the moment, it's very primitive and insecure, but…You'll need a valid mailbox for each active registry. These mailboxes are of form, if you read /var/spool/mail

```
cclite.<registryname>@domain.com.
```

If you use the more recent and preferred `read_pop_mail.pl` the (or much better `read_pop_mail_gpg.pl`) mailbox is the batch mailbox specified when you create the registry. Since this is read from the registry database, it should be OK in any cron.

What to Change

These are the items that to change at the top of `readmail.pl`.

```
# change to hardcoded lib path
use lib '../../../lib';

use Ccadmin;
use Cccookie;
use Ccu;
use Ccinterfaces;
use Ccconfiguration;

my $token;


# change to hardcoded path for configuration file,for example
# my %configuration = readconfiguration(/usr/share/cclite/config/cclite.cf);

my %configuration = readconfiguration();

#--------------------------------------------------------------
# change these two, if necessary

my $mail_dir = $configuration{'mailpath'};    # mail directory in readmail.cf
my %fields   = cgiparse();

# for cron, replace these with hardcoded registry name
# my $registry  = 'dogtown' ;

my $cookieref = get_cookie();
my $registry  = $$cookieref{registry};
```

Both mail and SMS payments use the main transaction routine in `Cclite.pm`. They both need to parse and translate the incoming textual data, which they do in a similar way. The payment format for mail is:

```
     Send x <currency> to <username> at <registryname> for
<reason phrase>
```

For example:

```
     Send 4 duckets to ddawg at dalston for barking
lessons
```

Both mail and SMS have *only been tested within one registry*, at present. SMS payments and other SMS transactions require the pin number as the first item of the transaction:

```
     p1234 Send 4 duckets to ddawg at dalston for barking
lessons
```

The from registry and to registry is assumed to be the same.

Mail payments will be recorded with a status of `waiting`, if this is not changed in the configuration file, `cclite.cf`. SMS payments are always recorded as waiting at present.
The person making the payment is recognised by the **`From:`** email address (and that's one of a number of insecurities). Therefore the **`From:`** address must correspond to the address in a valid, active user for that registry.

Currently the mailboxes for the registries are deduced from the active cookie as being cclite.<registry-name>. If using cron, the registry name must be hard-coded, see the comments in the code.

It's useful to run the script from the command line with some test email payments, to detect problems, before setting the cron. Lastly a `cron` job to run the script needs to be setup, either via Webmin or directly via `crontab`.

Installing RSS Small Ads as Cron

It has the same architecture as the other batch components and can run from the web control panel. It needs `userss=yes` in `cclite.cf` which in turn needs a require for `XML::Rss` . I've done this to keep the 'minimum' install somewhat simpler. It needs extra Perl modules to handle the RDF and XML parts, see the list at the front of the manual.

What to change

Here's what to change in `writerss.pl:`

```
BEGIN {

# for cron, you will have to supply the configuration file as a parameter to
readconfiguration

   # %configuration              =
readconfiguration('/usr/share/cclite/config/cclite.cf');

    %configuration        = readconfiguration();

    %configuration = (%configuration,%rssconfiguration) ;

}


use lib "$configuration{librarypath}";
use Ccu;
use Ccrss ;
use Cccookie ; # to get the registry token from the admin page...

my $token ;

# you'll have to hardcode these, if this is a cron

my $cookieref = get_cookie();

my $registry = $$cookieref{registry} ;

my $language = $$cookieref{language} ;
# these are the feed types, all ads, wanted ads, offered ads and matched ads, change this
to the feeds that you need

my @types = (all, wanted, offered, match) ;
```

Test on the command line, if possible by using:

```
./writerss.pl
```

and then set up and test a `cron`, to automate the feed process.

Installing Gammu style SMS payment as Cron

Mail and SMS payments have the same format and use the main transaction routine in `Cclite.pm`. They both parse and translate the incoming text data in a similar way. `readsms_from_gammu.pl` requires, `gammu-smsd` (see http://www.gammu.org/wiki/index.php?title=Gammu:SMSD), a compatible telephone (Nokia is often OK) and the right cable or a bluetooth dongle. The current testing was done with Huawei 1750 compatible dongle in the UK.

Gammu runs as a server via `gammu.sh` (in the batch directory) to pick up the messages and `readsms_from_gammu.pl` puts them into the transaction process. `readsms_from_gammu` can use SOAP and isn't necessarily on the same system as the cclite core.
The top of the script may need modification for the library and path to `gsmlib`, depending on where `gsmlib` is found on your system.

What to Change

These are `readsms_from_gammu.pl` lines to change to run as cron, normally it can be run from the Administration Control Panel without modification.

```
# hardcode configuration path, if running from a cron
my %configuration = readconfiguration();

# fixed needs testing
if ( !$configuration{'smslocal'} ) {
    use SOAP::Lite;
}

my $token;
my $file;

my $cookieref = get_cookie();
my %fields    = cgiparse();

# for cron: hardcode registry, cannot be read from web cookie
my $registry = $cookieref->{'registry'};
```

may need to be changed too.

As before, set up and test `readsms_from_gammu.pl` on the command line and then install. There's an argument for multiple numbers, since this would reduce the cost of incoming texts by sorting them by phone provider. SMS payments also have a standard title field at present.

If you stop using the phone, turn off gammu and readsms_from_gammu.pl !

Installing Batch payment as Cron

`readcsv.pl` provides processing for comma separated variable files (CSV(which can be uploaded via the web using the Admin control panel (which starts `ccupload.cgi`). These files can be used for manually recorded off-line payments or transfer of data from spreadsheets or other systems, for example.

Since this can be started automatically from the web control panel, there should be no real reason for running this as a cron now (11/2009).

As with SMS and mail, this can be run as a cron job. Change the location of the configuration in `readcsv.pl`:

```
# change the location of the configuration file here.
# needs to be hard coded because cron is run from, for example,
libexec
#
my %configuration ;
BEGIN {
%configuration =
readconfiguration('/home/yourdomain/domains/cclite.yourdomain.com
/trunk/config/cclite.cf') ;
```

Read about uploading the CSV file to the server in Batch Payment, that's how the data is supplied.

Installing/Enabling REST

A REST interface is now available. This is described in the REST chapter.

REST needs `AllowOverride` http://httpd.apache.org/docs/2.2/mod/core.html#allowoverride within the host or virtual host configuration usually `httpd.conf` if you're using Apache.

There’s enough to be useful but it's not complete (when is anything, ever complete?). I am currently using it to provide CURL http://curl.haxx.se/ based gateways to Drupal and Elgg. These prototype gateways are in the `gateways` subdirectory, *they are not production quality but are useful enough to be examples.*

The interface uses Apache rewrite rules in the web root (`public_html`) in the `.htaccess` file. Here's the example rule for paying somebody:

```
# REST style rewrite for Cclite
RewriteEngine on

# /pay/ddawg/dalston/23/duckets
RewriteCond   %{HTTP_COOKIE}  userLogin=(\w+).*token=(\S+)\b
RewriteRule   pay/(\w+)/(\w+)/(\w+)/(\w+)    /cgi-bin/cclite.cgi?
action=transaction&subaction=om_trades&fromregistry=$2&tradeSource=
%1&tradeDestination=$1&toregistry=$2&tradeCurrency=$4&tradeAmount=$3&token=
%2
```

This translates into an URL such as:

```
https://cclite.yourdomain.com/pay/ddawg/dalston/23/ducket
```

Also, the user must be logged in before any operation can be done. The login and logoff operation is also in this interface.

Installation Troubleshooting

Here are some of the commoner problems and their solutions.

If you have a specific problem, please ask for help on: http://groups.google.co.uk/group/cclite?hl=en then many people can benefit from the solutions.

Problem	**Possible Solution**
Internal Server Error	The scripts need to be world executable `chmod a+x` or ask your web space provider about this. Look in the Apache error log too!
REST interface doesn't work	The Apache configuration file `httpd.conf` may need to be changed to allow rewrite rules in `.htaccess`
Database upgrade to 4.0 or above needed	Your current MySQL installation needs to be at least 4.0
Database password or user name problem	There's probably a user or password problem when connecting to the database
Login failed problem	Please check the SERVER_NAME problem described in: http://groups.google.co.uk/group/cclite/browse_thread/thread/db65dfd305a5fc52?hl=en Also make sure that the hash per; module used SHA1 or SHA2 corresponds to the registry setup.
MIME:Base64=item encode_base64url decode_base64url problems	This seems to happen on Redhat. The MIME::Base64 module needs an upgrade.
Ccinstall: Cclite install Can't find install template directory: $dir/templates/html/ $language/install does not exist or unreadable?	Can't find the templates, these are needed for display. You need to check the path and permissions for the `templates` subdirectory against the configuration file and the directory layout on the server.

2. Administration

This part describes some of the initial and day-to-day administration for a Cclite installation. It mainly concerns web administration.

Overview

Cclite is mainly for use by users, on the web (with payment via email, SMS and batch upload). However, some communities may not have the web or use it partially, therefore a central administrator can do all the currency transfers as well as the administration. Whatever the administration and process model, the set up order is:

- Set up registry or registries
- Set up currency (ies) within registry
- Set up users or let them set up themselves
- Use the web control panel for batch jobs

Use by a Central Administrator

This is the more unusual case and requires a few modifications to the configuration:

- Remove the new user screen and link from public view
- Make all users set up as `active`, this can be done with `ccinstall.cgi`
- Make all payments `accepted`, , this can be done with `ccinstall.cgi`
- Remove all the user modification screens from public view
- Remove all the payment screens from public view
- Decide what to do about the yellow pages!

This gives a system where users can check their (and other peoples) balances on the web and the central administrator does the rest (add users, transfer currency, modify users etc:). There's also an intermediate option that users set themselves up and a central administrator activates them after induction; an identity check and other bits of process, for example.

Use by all Users

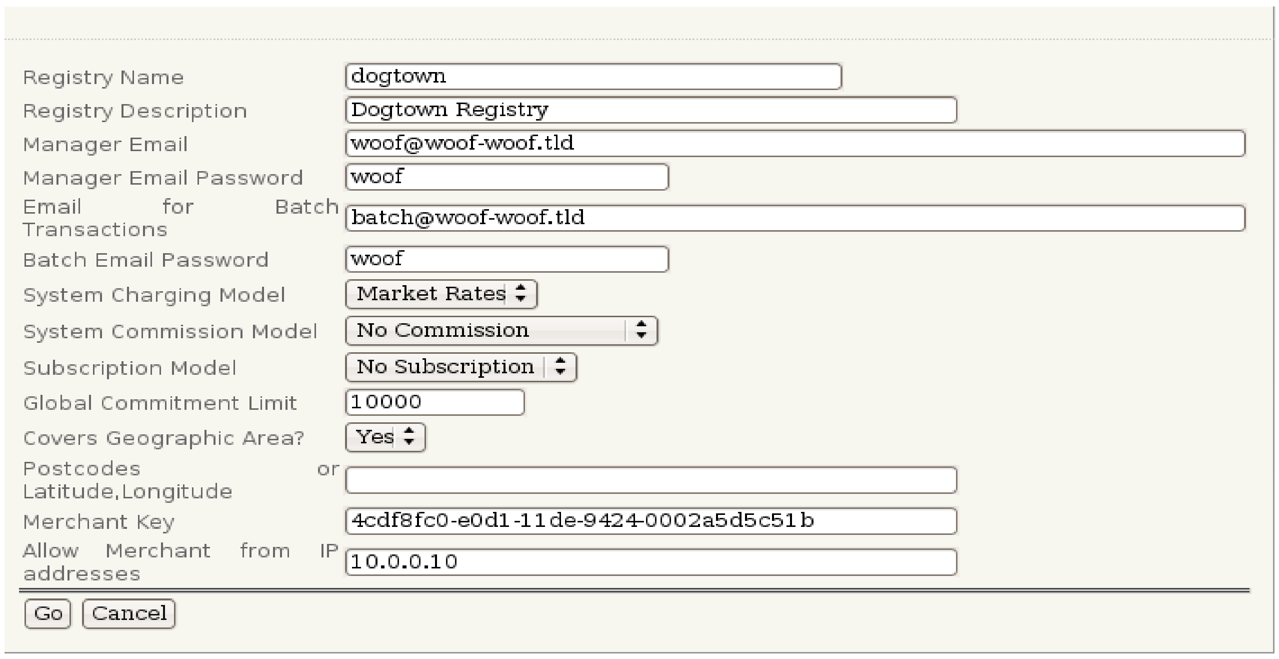
This is the expected mainstream use. Users set themselves up, confirm their email addresses to activate themselves and make their own transfers and create their own small ads. The central administrator only sets up registries (usually one registry for a given scheme, which they administer), currencies and partner registries, for example. The central administrator also applies the service charge, if it exists, since this isn't automatic currently.

Create Registry

This is the first operation after software installation. Normally; a registry is named after the scheme that it supports. For example, registry *dogtown* is for the LETS scheme in Dogtown.
This is done via `ccinstall.cgi`, a separate script. It's possible just to use `sql/registry.sql` via `mysql` for example.

If you're only planning to use one registry, remove the script afterwards. Otherwise take away execute permissions, if you leave it in place.

Look at `http://<yourdomain.com>/cgi-bin/protected/ccinstall.cgi,` link to the registry install:



Registry Name	dogtown
Registry Description	Dogtown Registry
Manager Email	woof@woof-woof.tld
Manager Email Password	woof
Email for Batch Transactions	batch@woof-woof.tld
Batch Email Password	woof
System Charging Model	Market Rates
System Commission Model	No Commission
Subscription Model	No Subscription
Global Commitment Limit	10000
Covers Geographic Area?	Yes
Postcodes or Latitude,Longitude	
Merchant Key	4cdf8fc0-e0d1-11de-9424-0002a5d5c51b
Allow Merchant from IP addresses	10.0.0.10

The global commitment limit is the maximum amount (currency independent) that a user can 'owe' in this registry. If present, it is tested before each transaction.

The postcodes or Latitude, Longitude field should contain comma separated lists. The merchant's key is for use with Tiki-Wiki, Joomla, osCommerce, Elgg and Drupal and needs to correspond to the merchant's key set on the other side of the gateway. In this example, a GUID: http://en.wikipedia.org/wiki/Globally_unique_identifier is used. This key permits straight through transactions from the shop and needs to be kept safely and changed if compromised.

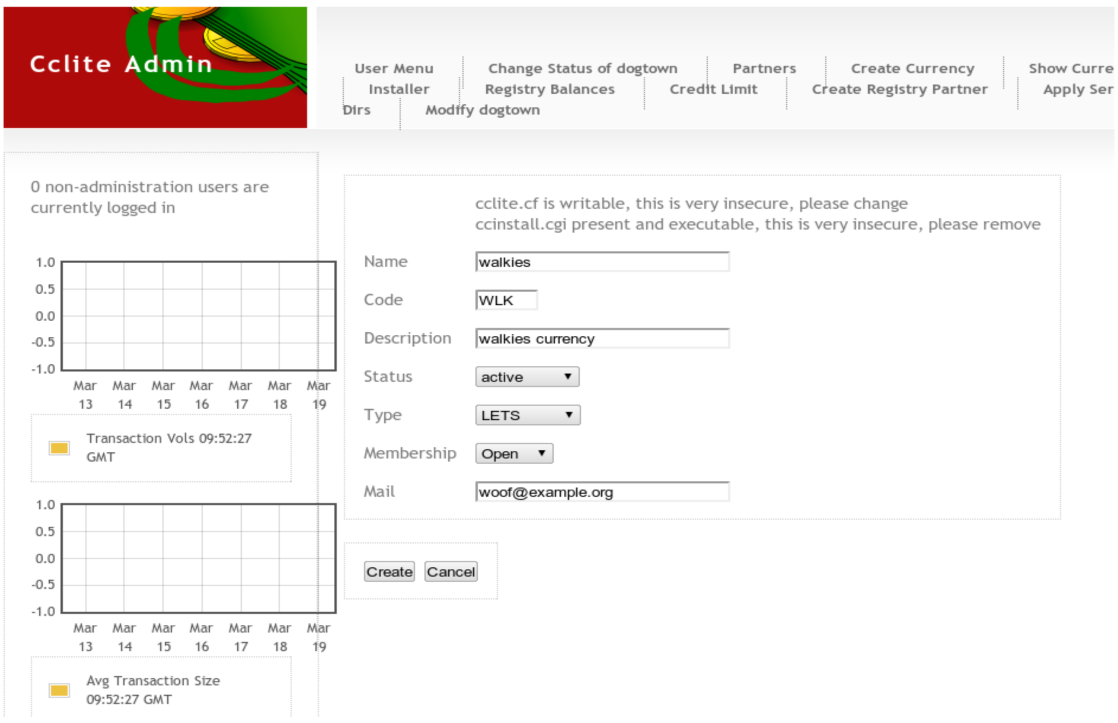
Dropping a Registry

There's no current way of doing this in Cclite. However, a registry is one database, so dropping the database with `phpmyadmin` or a command line tool will remove all traces of that registry. From 0.7.0 you'll need to remove the directories and files (input usually in `/var/cclite` and output usually in `/public_html/out`) associated with the registry too.

It's unlikely to make this an 'easy' operation, even if it's included (because these records will almost always need to be saved, for example, for tax purposes). I'll probably just provide a database name change to something that's agreed to be an inactive name.

Creating a Currency

First, a note, currencies *do not exist independently* from registries. They are part of a registry. This is a questionable decision for database structure but (I feel) a valid decision from the point of view of 'currency ideology'. The scheme or trading group (a group of *people*, as expressed by the registry component) is the primary thing and that gives rise to and contains a currency or currencies. Enough, already! A currency is set up via the **Create Currency** link.



0 non-administration users are currently logged in

🟨 Transaction Vols 09:52:27 GMT

🟨 Avg Transaction Size 09:52:27 GMT

cclite.cf is writable, this is very insecure, please change
ccinstall.cgi present and executable, this is very insecure, please remove

Name	walkies
Code	WLK
Description	walkies currency
Status	active
Type	LETS
Membership	Open
Mail	woof@example.org

A given currency is currently available to **all** members of a registry. There is a flag to ‘close’ a currency, that removes the currency from the drop-down lists. It's then up to the administrator to produce 'private' forms for the currency. This debate is also linked to map-type representations of groups, currencies, offers, wants etc. I'm planning to add a latitude and longitude field to the registry records, so that they can be represented as presence on a map.

Setting Up Users

Users set themselves up via the **New Account** screen in the general menu. A central administrator could also use this function, if the scheme is administered from a central point. Setting up users is described later in the manual. There's a new role of issuer, unused as yet, corresponds somewhat to Thomas Greco and Community Way models, where 'anchors' issue currency.

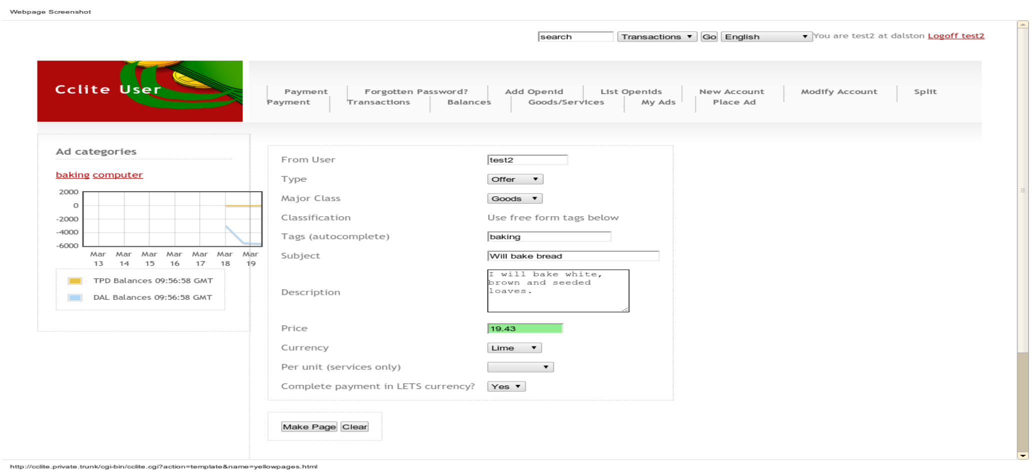
Goods/Services

As of version 0.9.0 there's free form tags to replace the fixed, english-only categories. This solves the previous problem of non-multi-lingual and fixed categories'

The RSS feed facilities also remain and there's a simple tag cloud for displaying categories of advert.

Adding and Offer or Want

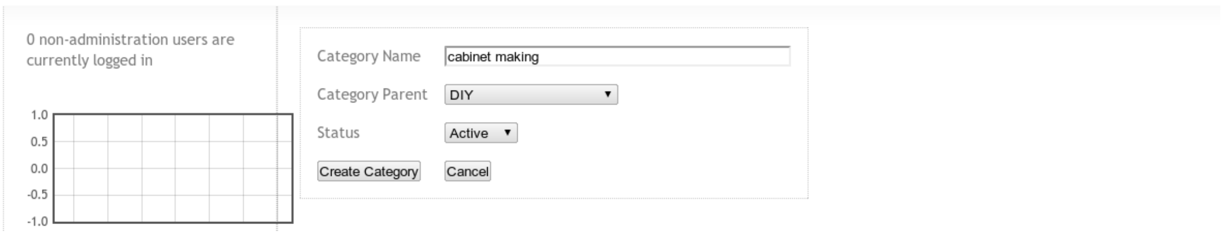
Click on the **Place Ad** tab and fill in the form. If the offer or want is completely payable in an alternative currency, choose the **Complete Payment** option. Adverts are added to the users personal pages in chronological order. A cron job to clear them, every three months (say) might also be useful.



From User	test2
Type	Offer
Major Class	Goods
Classification	Use free form tags below
Tags (autocomplete)	baking
Subject	Will bake bread
Description	I will bake white, brown and seeded loaves.
Price	19.43
Currency	Lime
Per unit (services only)	
Complete payment in LETS currency?	Yes

Adding a Category

This is currently fairly flaky. Just click on the **Create Ad Category** link and fill in the form. The category should then appear in the drop down for adding adverts. Don’t create millions of categories that only contain one thing! This is uncessary if free-form tags are used. Free form tages are preferred because they can be in any language.


0 non-administration users are currently logged in

Category Name: cabinet making

Category Parent: DIY

Status: Active

Customisation and Translation of Categories

Clearly, you may not want these categories or you may want them in another language. There's a CSV version of the English language categories in the `literals` subdirectory, it's called, `om_categories.csv`. There's a heading row, parent categories and child categories. You can use this layout to roll your own and then import them back into the `om_categories` table in the database. There's a sample below. You can also use phpmyadmin to get sql and change that etc.

```
"130";"1170";;"inactive";"USE OF FACILITIES"

"139";"1180";;"inactive";"FOR SALE"

"2";"1001";"1000";"active";"General Admin"

"3";"1002";"1000";"active";"Computer trouble shooting"
```

Linking Registries

This describes how to link Cclite registries so that payment can be made from an account in one registry to another. For example, someone from registry *islington* can move currency to *newham*. For this, the two registries are mutually linked (see below) and share a common currency (*this is simply a currency with exactly the same name, at present, there are no deeper semantics at present*).

In the alternative currency universe, trading between registries or schemes (sometimes called intertrading) is a subject that is full of controversy. Sometimes it's disallowed, sometimes it's allowed via a single system transfer account etc. etc. In Cclite, it could be done via a system account (because that is the simplest case of 'many' accounts) but also can be done between individual accounts. There's no *need* to use this, at all! I've built this into Cclite for example:

- It's nearly impossible to add as an afterthought
- It needn't be used in the majority of cases
- It needs experimentation, to check the computer science, usability, architectures etc.
- Reed's law and expanding number of trading systems
- I'm a believer in multiple differently scoped (regional, local) complementary currencies, this therefore fits somewhat with my personal model

I, personally, believe that the conventional currency problems are more linked to fractional reserve banking, derivatives, usury and money as a 'first class object' and not to this aspect.

It's also fair to say that *nobody knows*, since no-one has a *really big* alternative currency system in operation. Also, should any individual scheme be or become *really big* anyway?

Limitations and Comments

It's clear that this mutual linking won't scale into a large system. The next step, if it becomes popular is to provide a directory (could be a simple file, could be LDAP) of mutually accessible registries and currencies for a given region (say).I am unsure about this, because it may lead to central control by whoever is 'running' the directory. The computer science is not hard, the problems of governance, politics and control are more so.

On practical problem with linked systems is that they begin to display tidal effects of conventional money. That is, affluence washes from system to system enriching one and impoverishing another.

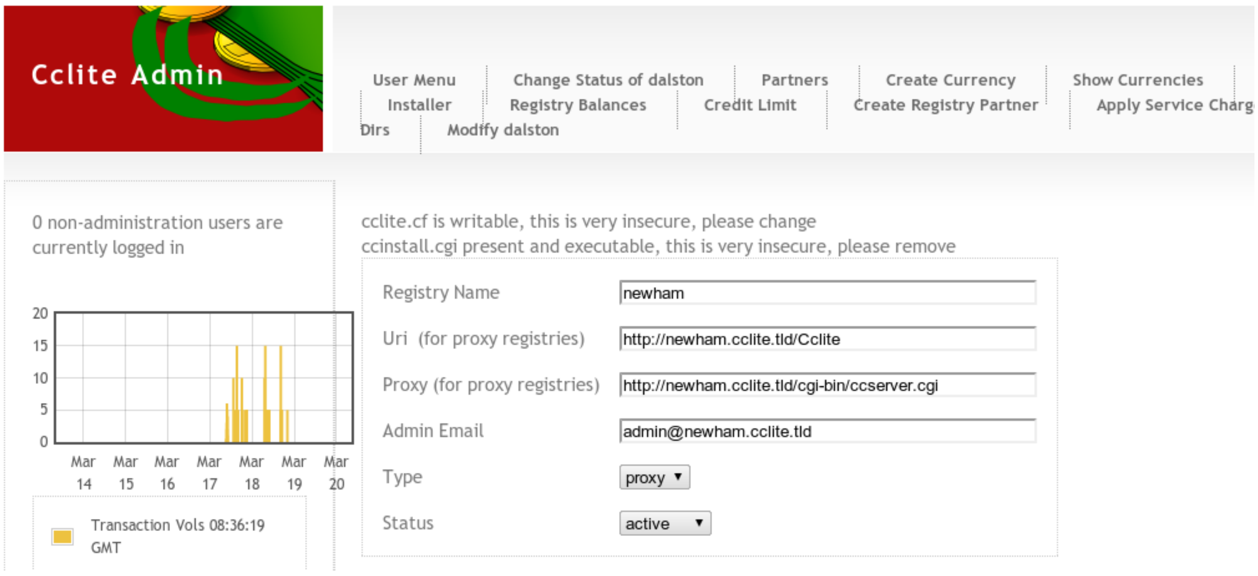
There's arguments for adding features for managing intersystem trade balances, perhaps via specialised 'exchangers', in future developments. I'm currently (November 2008) looking at Apache MQ [message queueus are a favourite in conventional finance] and Net::Stomp but we'll see...

Partner Registry

To enable this type of transfer, a registry needs a partner registry. The software must be configured as `multiregistry` in both cases.

Each registry must have its opposite number as partner (*islington* needs *newham* **and** *newham* needs *islington*). This also corresponds somewhat to the deontology (go on, surf now, you know you want to: http://en.wikipedia.org/wiki/Deontological_ethics !) of the situation.

This is set up in the administration as follows. Currently it's fairly insecure and both https and basic authentication for the SOAP exchanges can be used fairly easily, read the SOAP cookbook `http://cookbook.soaplite.com/` or contact me. If this gets used, further versions will probably have improved security as standard.



cclite.cf is writable, this is very insecure, please change
ccinstall.cgi present and executable, this is very insecure, please remove

Registry Name	newham
Uri (for proxy registries)	http://newham.cclite.tld/Cclite
Proxy (for proxy registries)	http://newham.cclite.tld/cgi-bin/ccserver.cgi
Admin Email	admin@newham.cclite.tld
Type	proxy
Status	active

Shared Currency

Both registries need a currency with exactly the same name (this is the only thing that makes the two currencies equivalent (be careful!)). This is set up in the administration screens using **Create Currency** as usual .

SOAP aspects

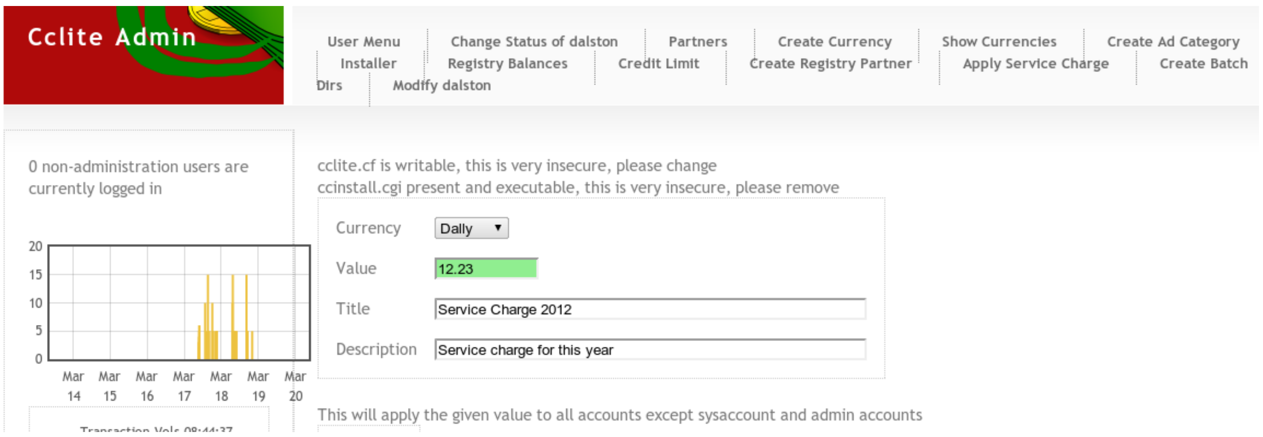
The `SOAP` transactions are transported via `http` (or `https`) in this version. This is the simplest because it doesn't require extra ports or create firewalling difficulties. I've also recently made some PHP clients for Cclite SOAP and discovered some difficulties associated with message format and passing hashes (these could just be my inexperience, but it's clear that it's 'hard' and therefore a bit inaccessible to people who 'just want it to work').

Therefore, from release 0.4.0 onwards, there's a REST (Representational State Transfer see http://www.xfront.com/REST-Web-Services.html which gives a quick summary of this). See the section on REST payment etc. in the rest of the manual to see how this is progressing.

For Amazon, about 85% of 'interfacers' use REST see http://www.oreillynet.com/pub/wlg/3005 suggesting that it's easier to understand and implement.

Apply Service Charge

It applies the service charge quoted to every non-manager non-sysaccount user within the given registry. It puts a balancing transaction for each into `sysaccount` in the registry. Remember that `manager` is used for software management, currency creation etc. and **should not participate** in day-to-day transactions. It operates as a standard transaction using the standard routine. There may be a need to condense the `sysaccount` entries into one single entry, for display convenience.



0 non-administration users are currently logged in

cclite.cf is writable, this is very insecure, please change
ccinstall.cgi present and executable, this is very insecure, please remove

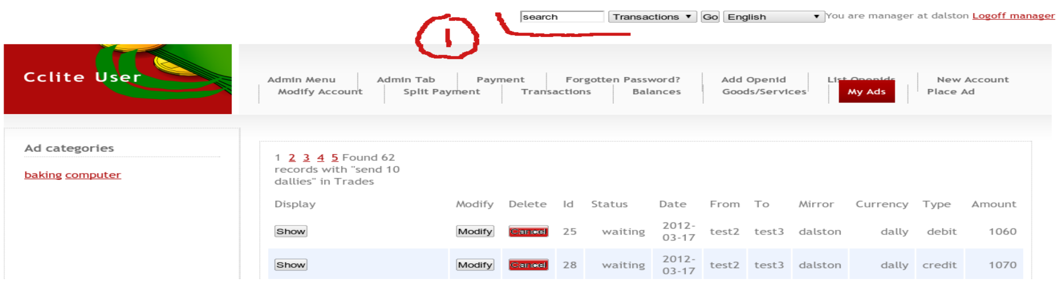
Currency	Daily
Value	12.23
Title	Service Charge 2012
Description	Service charge for this year

This will apply the given value to all accounts except sysaccount and admin accounts

There is also a `servicechargelimit` value in the configuration that can be used to prevent very large values being applied to the registry. The service charge is per currency not globally.

Cancelling Transactions

A manager can search for any transaction (via the search box) and cancel the transaction:



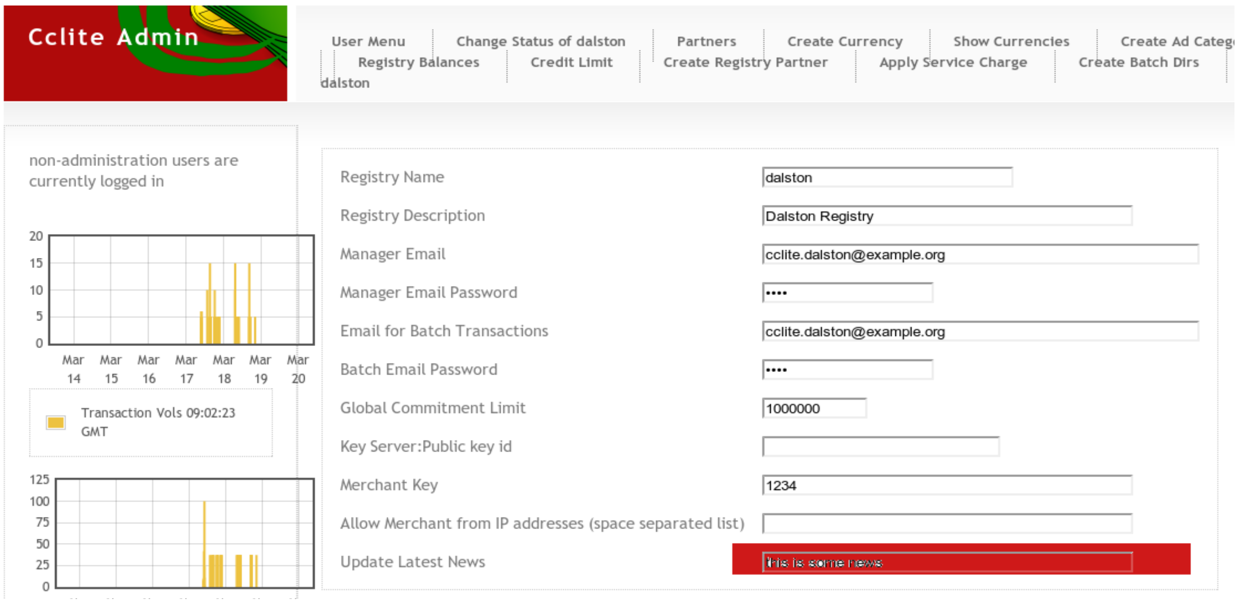
1 2 3 4 5 Found 62 records with "send 10 dallies" in Trades

Display	Modify	Delete	Id	Status	Date	From	To	Mirror	Currency	Type	Amount
Show	Modify	Cancel	25	waiting	2012-03-17	test2	test3	dalston	dally	debit	1060
Show	Modify	Cancel	28	waiting	2012-03-17	test2	test3	dalston	dally	credit	1070

This will modify the transaction status to cancelled and it will not be counted in volumes, balances etc. However, in the future, I believe that a registry with 'lots' (however we define that) of cancelled transactions should be regarded as dysfunctional.

Displaying News

This gives a one line text-only news display, use the field in your registry:



Registry Name	dalston
Registry Description	Dalston Registry
Manager Email	cclite.dalston@example.org
Manager Email Password	••••
Email for Batch Transactions	cclite.dalston@example.org
Batch Email Password	••••
Global Commitment Limit	1000000
Key Server:Public key id	
Merchant Key	1234
Allow Merchant from IP addresses (space separated list)	
Update Latest News	this is some news

To remove the current news, blank out the line.

Web Control Panel

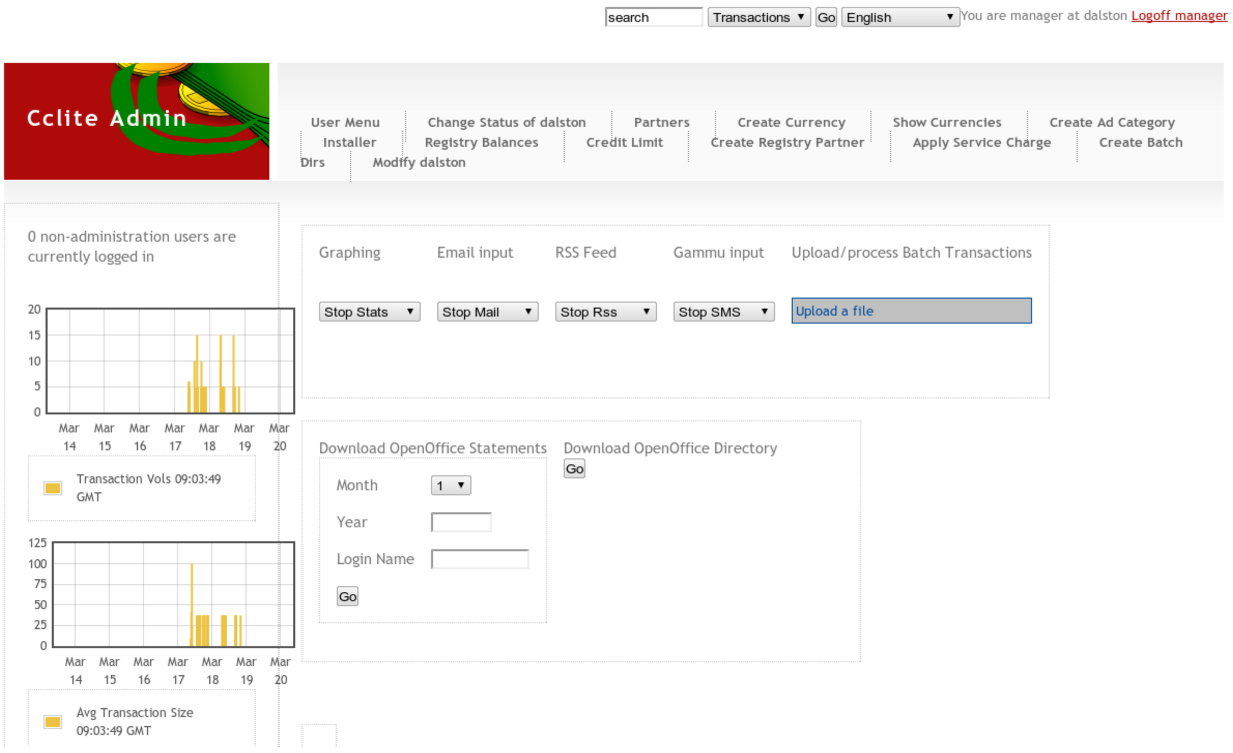
This control panel lets you to run the batch processes for reading email transactions, reading uploaded files, writing rss small ads and processing SMS transactions from Gammu. It also detects some of the problems with the batch directories and files needed for these processes.

It won't run the encrypted email collector: `read_pop_mail_gpg.pl` at present, because of an issue with ownership and access to the GPG key-ring

It also runs the process that produce the current example activity graphs, see below. The graphs as installed use Chart::Strip. You may have to do something with this manually, though.

Note: The best way to handle this page at the same time as user activity on the same system is to use two browsers. I use Firefox and Opera, for example. The web control panel page needs to stay (even minimised) so that the `Javascript setInterval` call which dispatches the batch jobs works. It can also be opened in a new tab via the **Admin Tab** *link* after **Admin Menu**

However, the whole set-up is a lot simpler for a 'web only' style installation, what most people have asked for.

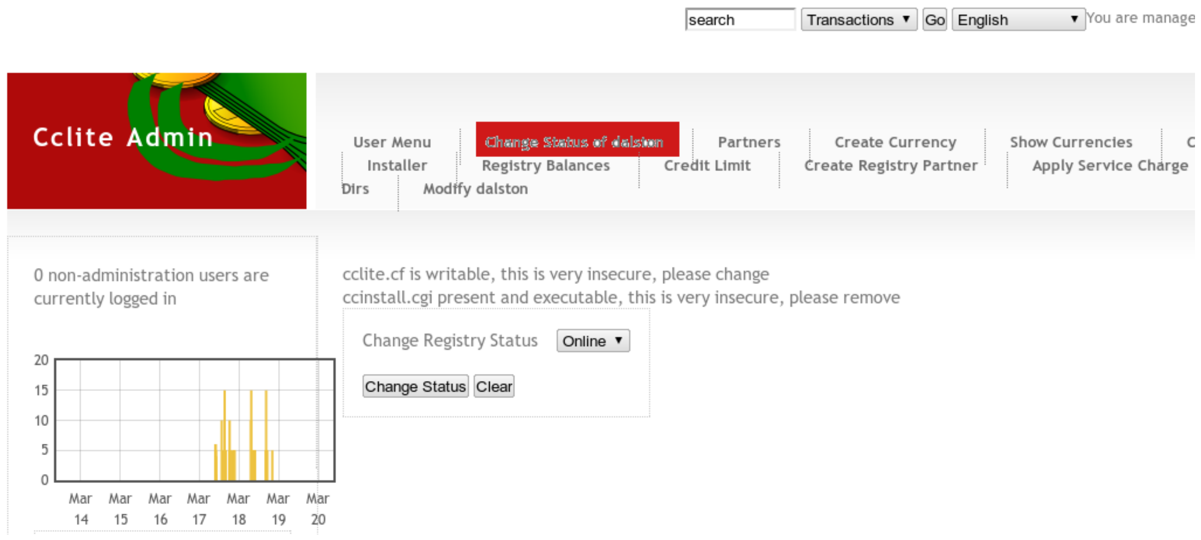


You are manager at dalston Logoff manager

0 non-administration users are currently logged in

Closedown and Reopen

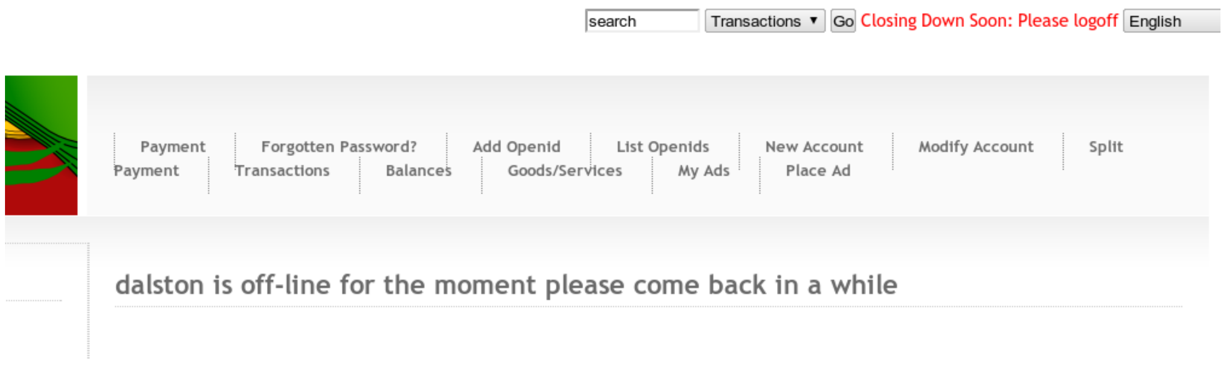
This isn't completely implemented yet. However after closedown, no-one except 'manager' can login. Also there's a message on the status bar for anyone that is currently logged in. It probably needs more logic to throw people off when the count of whos_online, implemented in Cccadmin.pm drops to 0 [the count does not include the manager. Since there's been a long time between releases, this is being released as-is. **Beware that currently, if people don't log-off cleanly they will stay in the count.**



0 non-administration users are currently logged in

cclite.cf is writable, this is very insecure, please change
ccinstall.cgi present and executable, this is very insecure, please remove

When a user tries to login, they get the following screen:



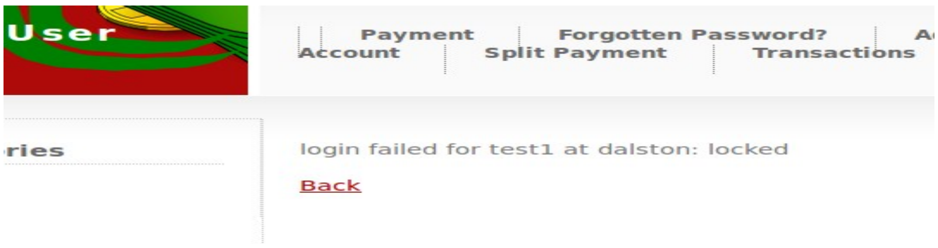
Closing Down Soon: Please logoff

dalston is off-line for the moment please come back in a while

Unlocking Accounts

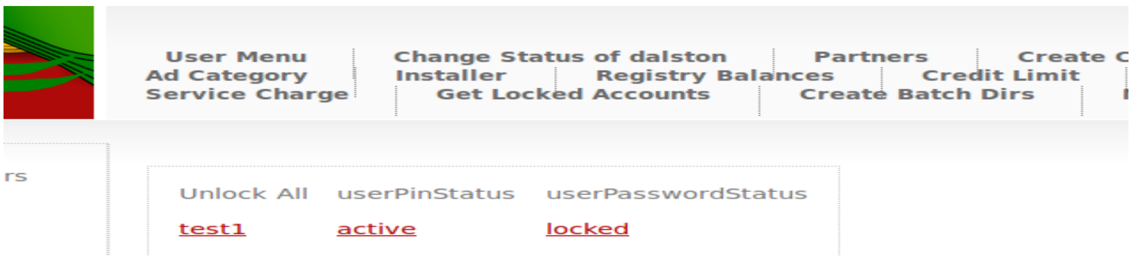
Accounts are now locked after 3 password failures or 3 pin failures for SMS transactions. This is part of a steady improvement in security.

So users may receive:



login failed for test1 at dalston: locked

They need to contact the manager who has an unlock facility for both passwords and SMS PIN numbers:



Unlock All	userPinStatus	userPasswordStatus
test1	active	locked

The password and PIN can be unlocked individually or both are unlock at once, by clicking on the user name in the first column.

3. Payment

This, of course, is the central function for Cclite. Payment can be web, mail, SMS or batch. Payment can be made within a registry or to a partner registry that shares a currency. To set up partner registries, see the Administration chapter.

Also, new as of 0.7.0, within payment, the manager (or a `userLevel admin` user) is allowed pay cash in and out and make payments between two users. This expanded payment page appears instead of the usual trades page for an administrator.

Payment can take place successfully or can fail with an explanatory message. Amongst the reason for failure are:

- Non-existent payee, check the spelling, or use auto-suggest for local transactions
- Over commitment limit, if there is one
- Problem at the remote registry, if using linked registries
- Various problems filling in the payment form

Of these, the problem with a remote registry may not lead to a diagnostic message. This is another weak area.

If payment is successful, a transaction reference will be returned. This is a long list of letters and numbers that is unique to the transaction.

For the technically minded, it's a hash of the transaction data using the `SHA2` or `SHA1`. `SHA2` is preferable since `MD5` (and `SHA1`) are suspected of producing collisions (things with the same reference).

On screen, it looks like this:

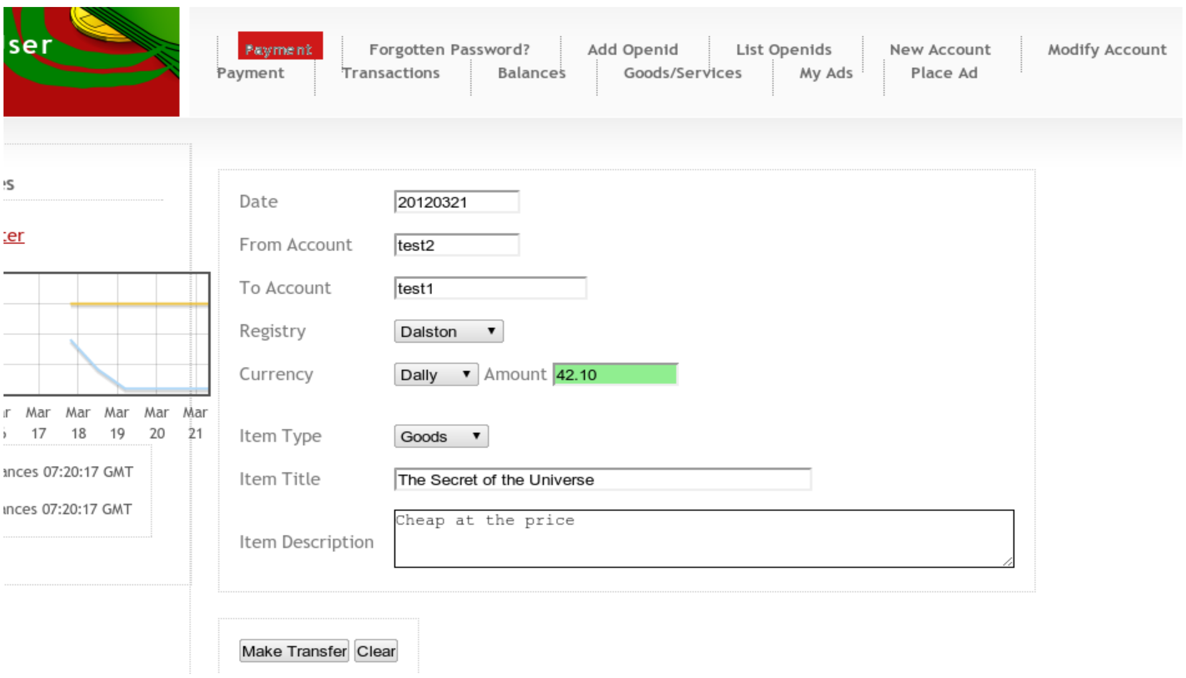
```
Transaction Accepted
Ref: YmH8Y7msvfZaPVWDCrljjmqF47bLcCiKV1zI4C7CEjHCmXkGQIH6b1jq1rkD8lbaplObJ6
lMcohApyR8Rc3pHQ
```

Web payment

There's a drop down list of suggestions for the destination account, if you're using a local registry. Just type the first one or two letters and wait a moment. If you don't want this, remove or change the permissions of `ccsuggest.cgi` this does all the auto-suggesting.

The web payment form is for the logged in user. Registries and currencies that are available are presented in two drop downs. At present the registries and currencies are independent lists (otherwise there must be an intermediate transaction, when the registry is chosen), therefore a transaction can still fail because a currency is not supported in a given registry.

If payment is made to a distant registry, there may be a pause whilst the SOAP transaction completes (depending on the general environment for the distant registry too).



Date	20120321
From Account	test2
To Account	test1
Registry	Dalston
Currency	Dally Amount 42.10
Item Type	Goods
Item Title	The Secret of the Universe
Item Description	Cheap at the price

The **Registry** drop down will list all registries that are set up as partners of the current registry or just the current, if the setup is single registry. The **Currency** drop down will list all 'open' currencies.

The administrator will have an unlocked **From Account** field and some extra cash style choices in the **Item Type** field, this is shown above.

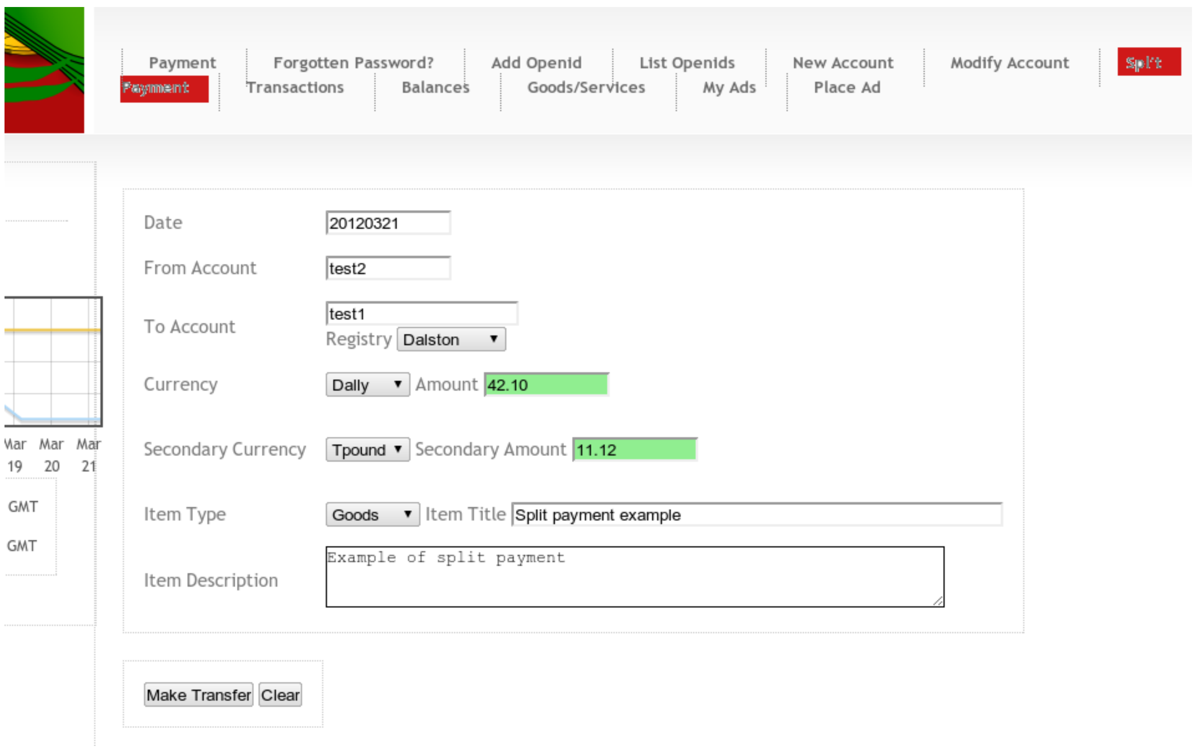
Split Web payment

This function is designed to aid systems and schemes that want to make payment for a single product or service in two different currencies. The most obvious example, is partial payment in national currency and partial payment in a community currency.

Both are recorded, within a registry using the same transaction reference (the transaction reference of the primary currency transaction).

The split web payment form is for the logged in user. The transaction is very similar to web payment (including the auto-suggest), but there are two drop downs for currencies and quantities. At present the registries and currencies are independent lists, therefore a transaction can still fail because a currency is not supported in a given registry.

If payment is made to a distant registry, there may be a pause whilst the SOAP transaction completes.

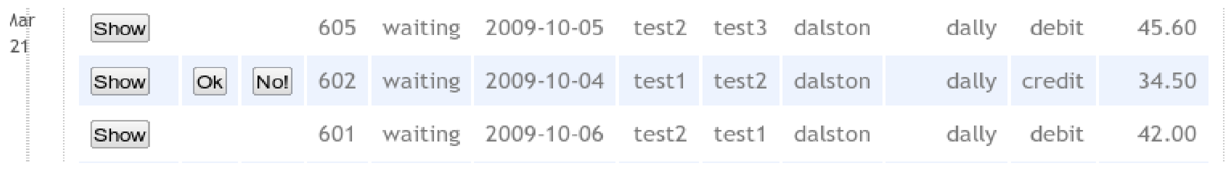


Date	20120321
From Account	test2
To Account	test1 Registry Dalston
Currency	Dally Amount 42.10
Secondary Currency	Tpound Secondary Amount 11.12
Item Type	Goods Item Title Split payment example
Item Description	Example of split payment

The **Registry** menu lists all registries that are set up as partners of the current registry. The **Currency** menu lists all 'open' currencies.

Waiting for approval

By default, a completed payment appears in the partner account as a `waiting` transaction. The receiving user clicks on the `OK` button in the transaction list to complete the transaction (or declines it if not wishing to receive payment).



Aar
21

Show			605	waiting	2009-10-05	test2	test3	dalston	daily	debit	45.60
Show	Ok	No!	602	waiting	2009-10-04	test1	test2	dalston	daily	credit	34.50
Show			601	waiting	2009-10-06	test2	test1	dalston	daily	debit	42.00

This may not be useful in centrally run systems. To switch this off, see the installation chapter, it needs a change in `cclite.cf: initialpaymentstatus=waiting to accepted` (using the installer). This means that a transaction is accepted by default when input. If a registry has 'lots' of declined transactions that is a problem for the registry. This may be reflected in one of the graphing tools, later on.

Mail Payment

NOTE: This is insecure at present. It is also single-registry, in spite of the extended syntax. The best method is to use `read_pop_mail_gpg.pl` and encrypt the mail payments. This is described under the heading Encrypted Mail Payment.

For the older method (`readmail.pl`) mail payment is made towards a queue belonging to the user's home registry, for example cclite.dalston@cclite.bigwavheuristics.com
The newer method using POP mail and the mailbox specified in the registry record is preferred. This mailbox is set up by hand, at present and mail pickup (`read_pop_mail.pl`) and processing uses `Net::POP3` at present and a fairly crude message parser (since `Mail::Message` etc. are exceptionally heavy and probably not installed on commodity hosts). The payment is from the mail address in the user's record, otherwise it will fail. This interface is insecure and experimental, at present. The payment format for mail is, on one line as *plain text* (although commodity email will probably do `multipart/alternative)` to ease parsing.

```
    Send x <currency> to <username> at <registryname> for
<reason phrase>
```

For example:

```
send 4 duckets to ddawg at dalston for barking lessons
```

Mail payments also have a standard title fields, at present.

Encrypted Mail Payment

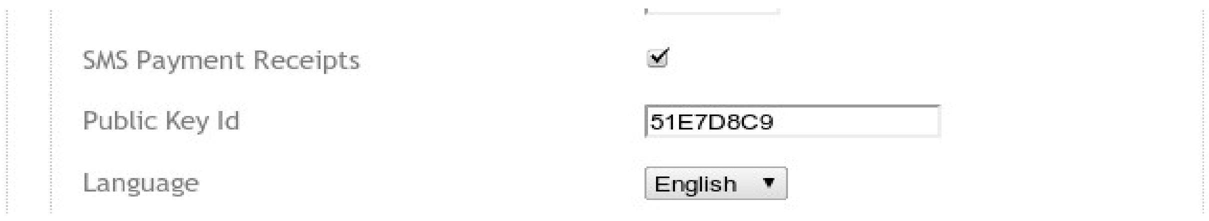
GPG Key and Cryptography knowledge

If you don't know anything about public key cryptography, then you need to do some reading, first of all:
http://en.wikipedia.org/wiki/Public-key_cryptography
http://en.wikipedia.org/wiki/GNU_Privacy_Guard

You need to read about GPG, the second URL too, this is the currently used cryptography package:
http://www.gnupg.org/ The options, for example, where to look for keys etc. are described here:
http://www.gnupg.org/documentation/manuals/gnupg-devel/GPG-Options.html#GPG-Options

The webserver system needs a key pair (public key for encrypting/signing the emails sent from it, private key for decrypting emails send using its public key).

Each cclite user will need a key pair (private for signing, public to the cclite server keyring or accessible key server for verifying signatures, this bit is not very developed at present). The public key id is a new field in the user record. At present, all the keys were on the server keyring when testing this, see the general and options link above to understand how to use a public keyserver, for example.



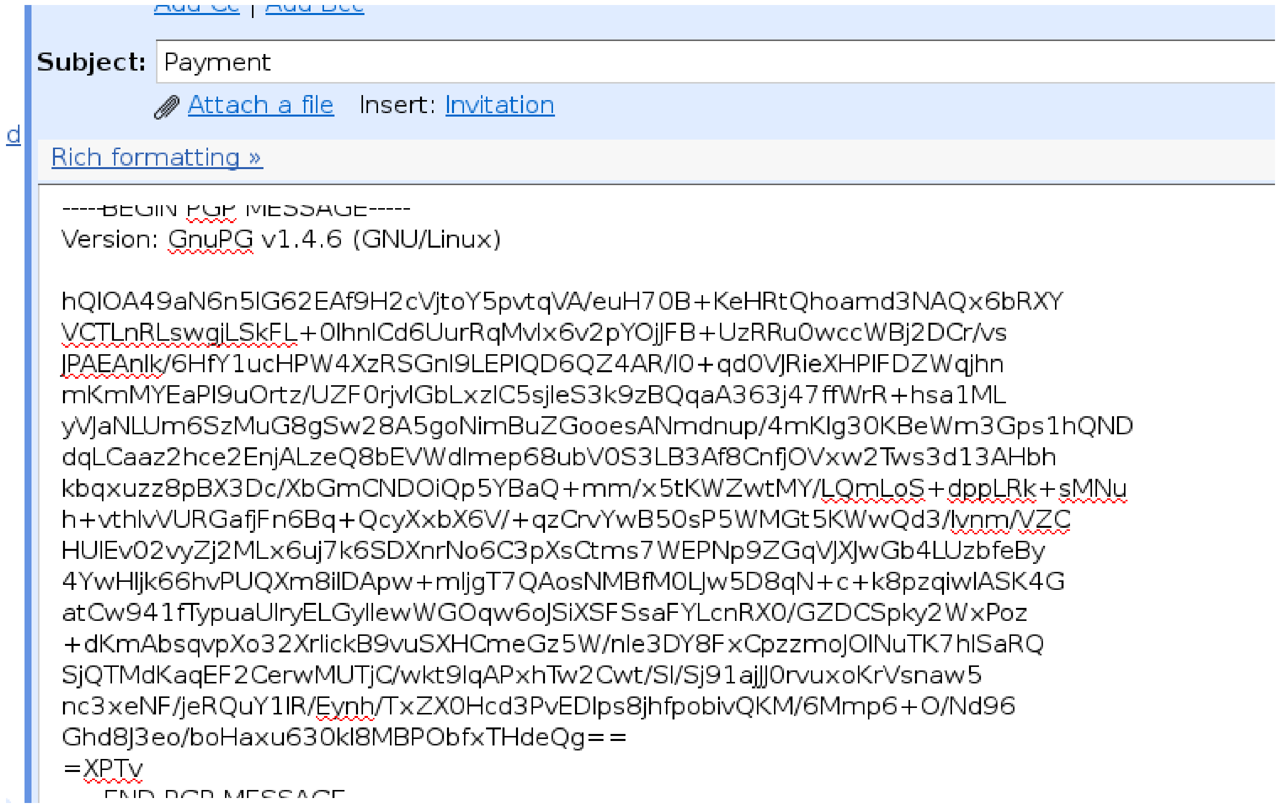
SMS Payment Receipts	✓
Public Key Id	51E7D8C9
Language	English

FireGPG Firefox Extension

I've used this for most of the testing, it's at: http://getfiregpg.org/s/home This extension allows you to encrypt from Gmail and some other web based mail services. Also there's a plugin for Thunderbird http://enigmail.mozdev.org/home/index.php but I haven't tried it.

Currently I've tried Gmail (good automatic integration) , Yahoo and Laposte.net (some manual encryption, select and encrypt, necessary). Probably in a working system, you may need to be prescriptive about webmail and mail clients that are 'allowed' or face increasing complexity in parsing incoming mail, your choice.

Your signed and encrypted email should look something like this:



Payment

```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.6 (GNU/Linux)

hQIOA49aN6n5IG62EAf9H2cVjtoY5pvtqVA/euH70B+KeHRtQhoamd3NAQx6bRXY
VCTLnRLswgjLSkFL+0IhnICd6UurRqMvIx6v2pYOjJFB+UzRRu0wccWBj2DCr/vs
JPAEAnIk/6HfY1ucHPW4XzRSGnI9LEPIQD6QZ4AR/I0+qd0VJRieXHPIFDZWqjhn
mKmMYEaPI9uOrtz/UZF0rjvIGbLxzIC5sjIeS3k9zBQqaA363j47ffWrR+hsa1ML
yVJaNLUm6SzMuG8gSw28A5goNimBuZGooesANmdnup/4mKIg30KBeWm3Gps1hQND
dqLCaaz2hce2EnjALzeQ8bEVWdlmep68ubV0S3LB3Af8CnfjOVxw2Tws3d13AHbh
kbqxuzz8pBX3Dc/XbGmCNDOiQp5YBaQ+mm/x5tKWZwtMY/LQmLoS+dppLRk+sMNu
h+vthIvVURGafjFn6Bq+QcyXxbX6V/+qzCrvYwB50sP5WMGt5KWwQd3/lvnm/VZC
HUIEv02vyZj2MLx6uj7k6SDXnrNo6C3pXsCtms7WEPNp9ZGqVJXJwGb4LUzbfeBy
4YwHIjk66hvPUQXm8iIDApw+mIjgT7QAosNMBfM0LJw5D8qN+c+k8pzqiwIASK4G
atCw941fTypuaUlryELGyllewWGOqw6oJSiXSFSsaFYLcnRX0/GZDCSpky2WxPoz
+dKmAbsqvpXo32XrlickB9vuSXHCmeGz5W/nle3DY8FxCpzzmoJOINuTK7hISaRQ
SjQTMdKaqEF2CerwMUTjC/wkt9IqAPxhTw2Cwt/SI/Sj91ajJJ0rvuxoKrVsnaw5
nc3xeNF/jeRQuY1IR/Eynh/TxZX0Hcd3PvEDIps8jhfpobivQKM/6Mmp6+O/Nd96
Ghd8J3eo/boHaxu630kI8MBPObfxTHdeQg==
=XPTv
END PGP MESSAGE
```

Incidentally, it's worth varying the description a little, even if you are making regular payments, otherwise you are providing useful information depth for replay and person [we mustn't say man] in the middle attacks. This is not the last word in secure payment but the combination of https and GPG isn't too bad.

The status of the transaction and receipt will also return encrypted [since it provides useful information about the transaction itself] and must be decrypted using FireGPG, for example.

SMS Payment

SMS payment is made towards a gateway mobile number corresponding to the home registry (and perhaps to the user's mobile provider, this reduces the payment cost).

Payment reception can either be done with a phone + Gammu or with a commercial external gateway. There's a lot more work on the Gammu + phone dongle facility in release 0.9.3. Set up and architecture for both is described later in the manual.

The payment must come from the mobile phone number that the user has registered in their user record. Registration in the user record is in international format, for example *44777755555* (UK number) or *33999911111* (French number).

These are the current formats accepted for payment messages, the keyword can be **send** or **pay**:

```
p123456 pay 5 to 447855524667

p123456 pay 5 to 447855524667 for other stuff

p123456 pay 5 to 447855524667 for stuff

p123456 pay 5 limes to test2

p123456 pay 5 limes to test2 for otherstuff
```

.
and in general, for example:

```
     p<pin-number> send x <currency> to <username> at
<registryname> for <reason phrase>
```

The message parsing is in large regular expressions in modules within `lib/Ccsms`. You should modify there to accept different messages. The regular expressions are also not internationalised at present, please use non-accented characters...

SMS works with any local registry and transactions have a standard title field at present. The pin number is set in the user record for the sending user.

Please see the SMS chapter later for set up and a current list of all SMS messages.

Payment via REST

To pay someone some amount, provide a sequence of URLs as the example below:

```
http://cclite.yourdomain.com/logon/ddawg/chelsea/<password>
http://cclite.yourdomain.com/pay/mmouse/chelsea/23/ducket
http://cclite.yourdomain.com/logoff
```

Note that the currency name is the exact name (in this case singular). There is now processing to make english plurals singular, but it's probably not best practice to use plurals.. If accepted, it will reply with a json record for the transaction as receipt reference:

```
{
    "registry": "dalston",
    "table": "om_trades",
    "message": "OK",
    "data": [
        {
            "message": "Transaction Accepted",
            "reference": "zRZqe6A6Lzs5v6hNbLGfeI8Zlv8"
        }
    ]
}
```

Otherwise the transaction can be rejected for all the same reasons (non existent destination, currency etc.) as a form based or any other transaction.

This can form the basis for Perl LWP and PHP libcurl (for example) remote client programs. There is a an example (but rudimentary and minimally tested!) `curlclient.php` in the php subdirectory. The Elgg and Drupal experimental interfaces also use `curl` and the REST interface.

Batch Payments

Batch payments are processed via a batch file. The contents of the batch file should be as below (lines starting with # are skipped and space lines are skipped).Please note the date format, also:

```
#date,from,to,fromreg,toreg,currency,type,taxflag,value,desc

04/10/2009,"test1","test2","dalston","dalston","dally","debit","N",34500,"val
id but big"
05/10/2009,"test2","test3","dalston","dalston","dally","debit","N",45a,"inval
id quantity"
06/10/200b,"test2","test1","dalston","dalston","dally","debit","N",42,"invali
d date"
04/10/2009,"test1","test2","dalston","dalston","dally","other","N",345,"not a
debit"
05/10/2009,"test2","test3","dalsto","dalston","dally","debit","N",456,"invali
d registry"
06/10/2009,"test2","test1","dalston","dalston","daly","debit","N",42,"invalid
currency"
04/10/2009,"testrrr","test2","dalston","dalston","dally","debit","N",345,"unk
nown from"
05/10/2009,"test2","testrrr","dalston","dalston","dally","debit","N",456,"unk
nown to"
06/10/2009,"test2","test1","dalston","dalston","dally","debit","L",42,"bad
tax flag?"
04/10/2009,"test1","test2","dalston","dalston","dally","debit","N",34.50,"val
id"
05/10/2009,"test2","test3","dalston","dalston","dally","debit","N",45.60,"val
id"
06/10/2009,"test2","test1","dalston","dalston","dally","debit","N",42.1,"vali
d"
06/10/2009,"test2","test1","dalston","dalston","dally","debit","N",22,"valid"
```

This file is uploaded into into the csv directory (defined by `csvpath=` , in cclite.cf and the registry name) using the **Upload Batch Transactions** function in the **Admin Menu**.

It is processed by the `readcsv.pl` which is started atuomatically at the end of the upload, a file `xxxx.csv.out.<timestamp>` is written into `public_html/out/csv/<registry-name>` as an html file containing the processing results (so results are available in the web root. This directory can be .htaccess protected or just with allowed index, depending on the need or not for security.

The input file is renamed to `xxxxx.csv.done.<timestamp>` so it is not reprocessed.

4. OsCommerce Gateway

You ***DO*** need to know a little bit about OsCommerce™ itself to use this! This hasn't been worked on for quite a while, so your mileage may vary.

Philosophy and Structure

This describes the Cclite payment gateway for OsCommerce™. OsCommerce™ needs to be used in a slightly different way to support a social credit system or a closed barter group.

For example, there will be many producers and many consumers and they will probably be the same people. Also, a payment may not be to one producer but may be split up amongst many (for example, a rural group, A supplies eggs, B supplies bread but all this is accumulated into one 'order').

Therefore the gateway uses the `manufacturers_name` field in the `manufacturers` table in OsCommerce™ to identify 'producers' who are to be paid within the corresponding Cclite system. The manufacturer name is the same as the `userLogin` name in the `om_users` table in Cclite.

Sooner (or later, or you can do this!) I'll supply a script to create manufacturers for every registered Cclite user (or should that be the other way around?).

The gateway is a standard OsCommerce™ gateway, written in PHP and installs using the install within OsCommerce™.

Restrictions

This is an alpha version of the gateway (still! as of 2009)

At present, one Cclite registry is connected with one shop front and only one currency is allowed per shop. OsCommerce™ handles conventional money where all the different currencies are linked via exchange rates. For many alternative money theorists, exchange rates and conversion are often undesirable (because they leads to pure monetary operations, currency futures and derivatives, for example), so I don't plan to work in this area of the software at present. Currencies are independent in Cclite, so it doesn't make to much sense to alter the philosophy here.

Also, I did not (and do not) want to produce a specialised and forked version of OsCommerce™ (this usually makes life more difficult for everyone, OsCommerce folks, myself and the rest of the universe).

Installation Overview

To install and use the gateway, do the following steps:

1. Generate a **Merchant Key** for the Cclite registry that is to be connected.
2. Currently, use http://www.fourmilab.ch/hotbits/generate.html or a guid http://en.wikipedia.org/wiki/Globally_Unique_Identifier to provide a key. This needs to be the same on the Cclite side and on the Oscommerce side.
3. This is used as a password between the shop and the registry. It corresponds to the API key in a standard credit card gateway.
4. Install and configure the Cclite gateway within OsCommerce administration. This is a standard OsCommerce gateway install.
5. Put the **Merchant Key** into Cclite registry that is to be connected, using the **Modify <name-of-registry>** in the **Admin Menu**.
6. Set up `manufacturers` corresponding to Cclite `users` with the OsCommerce shop administration software.
7. Set up products for each `manufacturer`.

Generating a Registry Merchant Key

NOTE: This will soon require `openssl` or `gpg` to be installed on the Cclite machine, the hotbits key or guid is an alpha-version compromise.

You can just choose a ‘weak’ key (1234, for example!) if you are testing or experimenting.

Install and Configure the Cclite Gateway

This is a manual install for the present.

1. Copy `cclite.php` from the gateways subdirectory of Cclite into `catalog/includes/modules/payments` within OsCommerce

NOTE: Yes, there are two files called `cclite.php,` one in `modules` and one in `languages`!

2. Copy `cclite.php` from the gateways subdirectory of Cclite into `catalog/includes/language/payments` within OsCommerce
3. Go to the administration tool and click on **Payments**.
4. Choose **Cclite** and click on **Install**, see below.
5. Edit any of the fields:

Enable CCLITE Module	True	Needs to be true, otherwise this won't be offered as a payment method
Transaction Currency	Only ducket	Name of Currency to be accepted
Currency Symbol	DCK	This needs to correspond to the currency symbol used in Cclite
Payment Zone	--none--	Not used at present
Set Order Status	Default	Not used at present
Sort order of display.	0	Sort order for these parameters, not used

Merchant Key	1234	Shop key that is generated in Cclite or user chosen key. Attention: a 'weak' key will mean that others may be able input transactions. Change this key fairly frequently.
Gateway URL	https://yourdomain.com/cgi-bin/cclite.cgi	URL of the Cclite installation that is being used to collect payments
Default Registry	chelsea	Registry used to collect transactions
Enable Payments	live	Can be 'live' or 'test'. With test, Cclite will display the data it's receiving without finishing the transaction.

Testing the Gateway

To see whether the shop is communicating with the gateway, set **Enable Payments** to `test` and check the output from Cclite. Payments are shown by manufacturer and each one will appear as a separate cgi value on the Cclite screen. When testing is finished, re-configure the gateway to `live`.

Removing the Gateway

Just click **Remove** and the gateway is removed from the shop but can be reinstalled at any time.

5. REST Interface

See the Install chapter for the configuration required for REST.

I currently think that this type of interface will be easier to use than the SOAP interface (which requires much tighter coupling between the client and server).

The REST operations that are currently supported are shown with commentary below. These operations are also a basis for curl-based: http://curl.haxx.se/ interfaces used for interfacing with Elgg and Drupal and Tiki-Wiki. These are very rough at the moment and are in the `gateways` subdirectory.

The design I have used when working with these is stateless, inteface logs on, does something and logs off. This may be inefficient, but there's other stuff to worry about before we get to that.

Also currently, the default return mode is changed to json this is the start of a multi-representation interface, making for easier and neater use with linked pieces of software. This change was made to support the rough connectors for Elgg , Drupal and Tiki-Wiki. The proposed allowed modes are json [default], csv and html.

Please look at `public_html/.htaccess` to see, in detail, how these urls are re-written.

OpenTransact

As OpenTransact: http://www.opentransact.org/ matures, the fields in the REST part, at least will show some pcompatiblity with it. Certainly Cclite will get Oauth2, sooner rather than later.

Payment

You need to logon before you can do this, either as a user or by presenting a time dependent hash of the merchant key from a known IP address.

http://cclite.yourdomain.com//pay/ddawg/chelsea/23/luckets

This will reply with accepted and a hash of the transaction as reference:

```
{
    "registry": "dalston",
    "table": "om_trades",
    "message": "OK",
    "data": [
        {
            "message": "Transaction Accepted",
            "reference": "zRZqe6A6Lzs5v6hNbLGfeI8Zlv8"
        }
    ]
}
```

This is also described in the Payment Chapter.

Credit, Debit, Demurrage

These are not currently used and there's no supporting code in these cases.

```
# /credit/ddawg/dalston/23/duckets
# /debit/ddawg/dalston/23/duckets
# /demurrage/ddawg/dalston/23/duckets
```

Logon a User

Logs a user on using the merchant key/password for the connector rather than the individual password.

http://cclite.yourdomain.com/logon/ddawg/dalston/password

Add a User Directly

This enables the Drupal connector to add a 'stub' user in Cclite whenever a Drupal user is added. It will add 'auto-created' into the description. As of release 0.9.0 this will only work from IP addresses that are in the allowed list in the registry record.

http://cclite.yourdomain.com/direct/adduser/dalston/test1/email@dddd/

```
{
"registry": "dalston",
"table": "om_users",
"message": "OK",
"data": [ ]
}
```

This is be `NOK` if the user add fails, usually duplicate userLogin or userEmail

Modify a User's Email Directly

This enables the Drupal connector to modify a Cclite user's email when the Drupal entry is modified. As of release 0.9.0 this will only work from IP addresses that are in the allowed list in the registtry record and with a valid merchant key.

http://cclite.yourdomain.com/direct/modifyuser/dalston/test1/email@dddd/

```
{
"registry": "dalston",
"table": "om_users",
"message": "OK",
"data": [ ]
}
```

This is be `NOK` if the user add fails, usually duplicate userLogin or userEmailLog the current user off, completes a session in a connector, for example.

Logoff

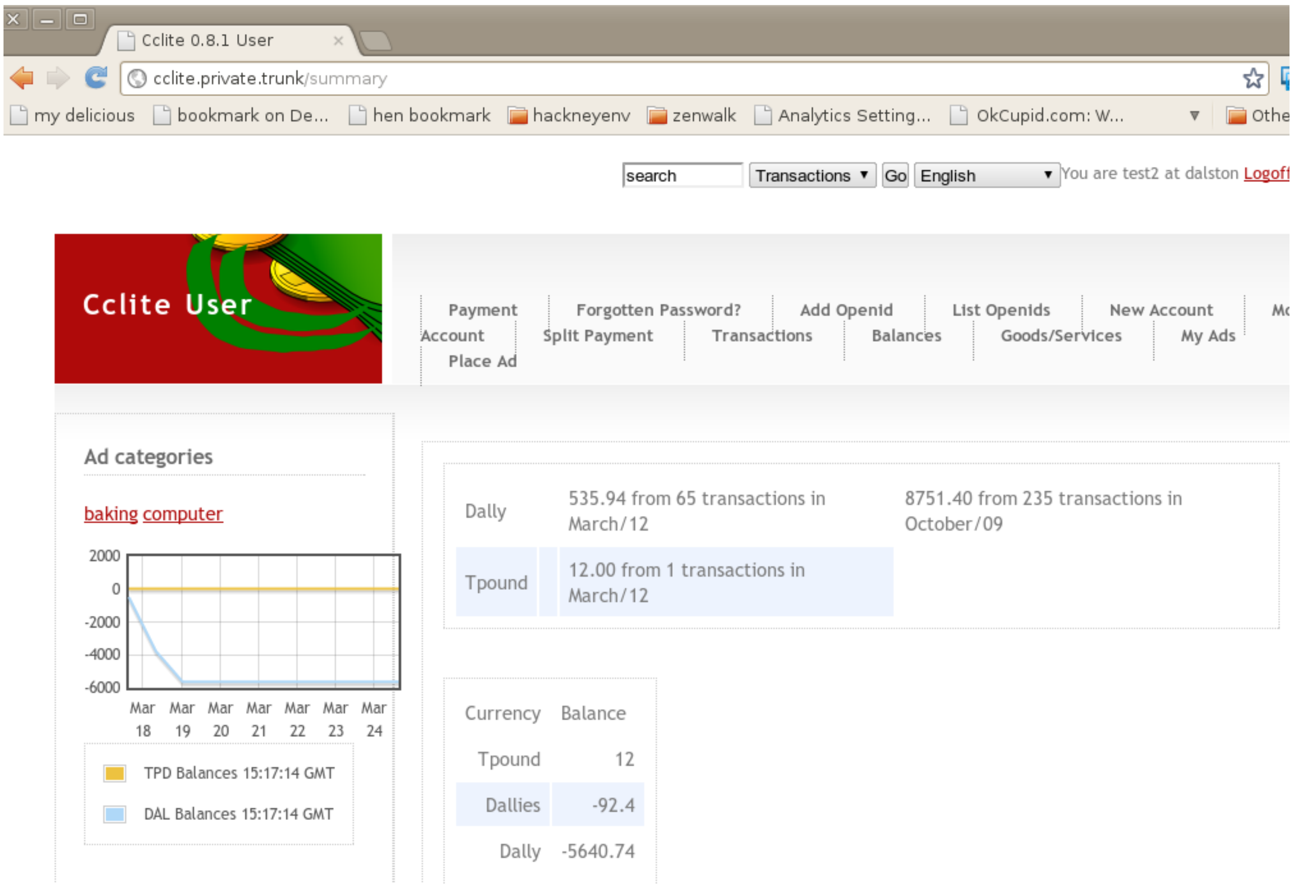
http://cclite.yourdomain.com/logoff

Trading Summary

This will currently only deliver a summary for the logged in user, but the syntax allows extensions to queries by other users. Also can deliver json, as in second syntax.

http://cclite.yourdomain.com/summary

http://cclite.yourdomain.com/summary.json



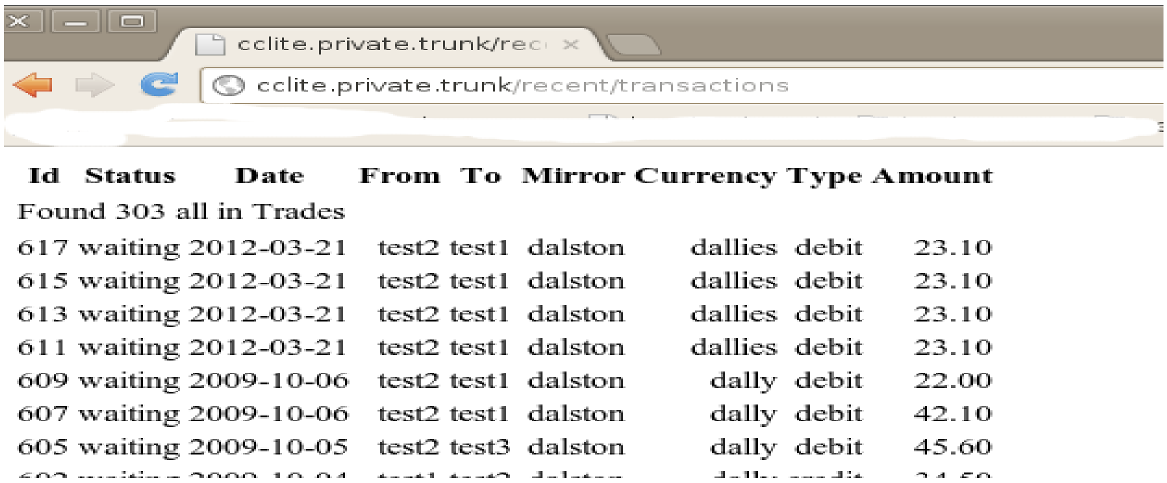
Dally	535.94 from 65 transactions in March/12	8751.40 from 235 transactions in October/09
Tpound	12.00 from 1 transactions in March/12	

Currency	Balance
Tpound	12
Dallies	-92.4
Dally	-5640.74

Detailed Transactions

This is not ideal at present because it'll just deliver the first page of a paged set of transactions in either csv[ish] as default or json:
http://cclite.yourdomain.com/recent/transactions
http://cclite.yourdomain.com/recent/transactions.json

http://cclite.yourdomain.com/ddawg/transactions/nnn
http://cclite.yourdomain.com/ddawg/transactions/nnn.json



```
 Id  Status     Date     From  To  Mirror Currency Type Amount
Found 303 all in Trades
617 waiting 2012-03-21  test2 test1  dalston    dallies  debit    23.10
615 waiting 2012-03-21  test2 test1  dalston    dallies  debit    23.10
613 waiting 2012-03-21  test2 test1  dalston    dallies  debit    23.10
611 waiting 2012-03-21  test2 test1  dalston    dallies  debit    23.10
609 waiting 2009-10-06  test2 test1  dalston      dally  debit    22.00
607 waiting 2009-10-06  test2 test1  dalston      dally  debit    42.10
605 waiting 2009-10-05  test2 test3  dalston      dally  debit    45.60
```

```
 {"registry":"dalston","table": "om_trades", "message":"OK",
"data": [
{"id": "592",
 "tradeSource":"test1",
"tradeDestination":"test2",
"tradeType":"credit",
"tradeDate":"2009-10-04",
"tradeId":"592",
"tradeMirror":"dalston",
"tradeStatus":"waiting",
"tradeCurrency":"dally",
"tradeAmount":"3450"},

{"id": "595",
 "tradeSource":"test2",
"tradeDestination":"test3",
"tradeType":"debit",
"tradeDate":"2009-10-05",
"tradeId":"595",
"tradeMirror":"dalston",
"tradeStatus":"waiting",
"tradeCurrency":"dally",
"tradeAmount":"4560"}
```

PHP Client for REST

There's a php client called `curlclient.php` for REST in the php subdirectory. It's been very minimally tested. You'll need to change at least the following lines, to try it out:

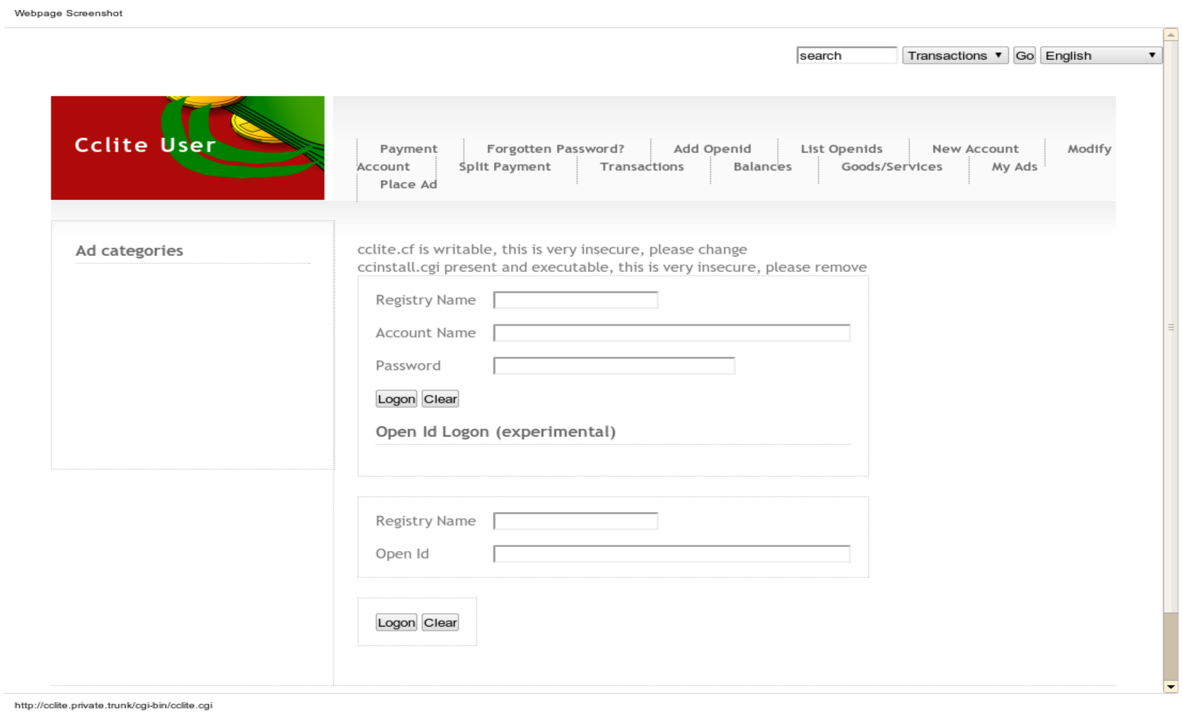
```
/* logon and store the logon cookies */

/* configuration variables */
$domain          = "yourdomain.com" ;
$cookiefile      = "/tmp/cclitecookies" ;
$user            = "youruser" ;
$registry        = "yourregistry" ;
$password        = "passwordofyouruser" ;
$currency        = "lucket" ;
$amount          = "youramount" ;
$payto           = "usertobepaid" ;
```

6. Logging In and User Information

Logging In

This is how the logon screen should look after a successful install.



cclite.cf is writable, this is very insecure, please change
ccinstall.cgi present and executable, this is very insecure, please remove

You can change this screen from the standard to give a drop down (`multiregistry`) or fixed value for the registry name by changing `logon.html` in the `templates` subdirectory.

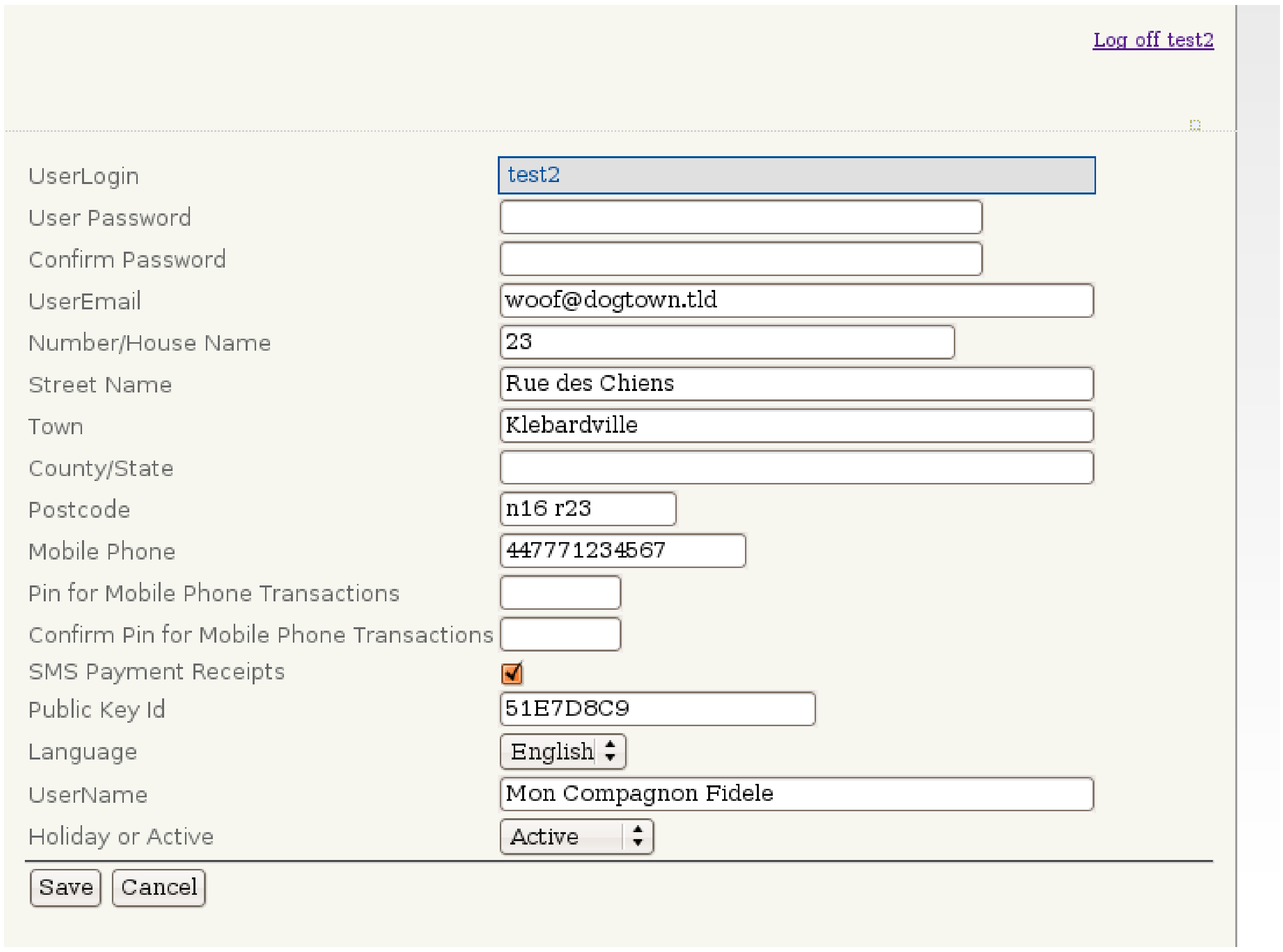
Password Locking

As of release 0.9.3, the account will lock after three incorrect password attempts. Then it needs to be unlocked by the manager.

Remember to remove or take away execute permissions from `ccinstall.cgi` and make `cclite.cf` read-only to make the messages go away. The function that produces this is called `install_grumble` and it lives in `Ccsecure.pm`. *Remember also, to change the default manager's password.*

Changing User Details

This is the current user modification screen. An administrator has control of more options, suspension, pre-delete etc. but cannot change the user password. If the password is corrupted, change with an email reset.



UserLogin	test2
User Password	
Confirm Password	
UserEmail	woof@dogtown.tld
Number/House Name	23
Street Name	Rue des Chiens
Town	Klebardville
County/State	
Postcode	n16 r23
Mobile Phone	447771234567
Pin for Mobile Phone Transactions	
Confirm Pin for Mobile Phone Transactions	
SMS Payment Receipts	✓
Public Key Id	51E7D8C9
Language	English
UserName	Mon Compagnon Fidele
Holiday or Active	Active

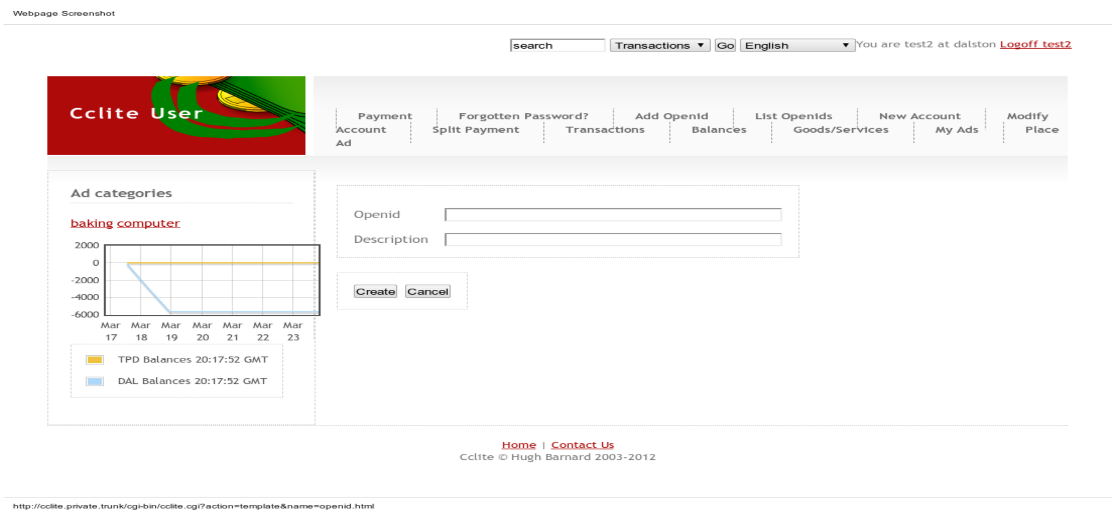
Note also, SMS payment receipts which often have a cost in national currency can be switched off and on. By default they are off, when a user is added to the system.

Using OpenID

This is an experimental feature from version 0.8.0. At a later stage Oauth will probably be added to give a generalised server-style authentication interface. For a general description of OpenId see: http://openid.net/

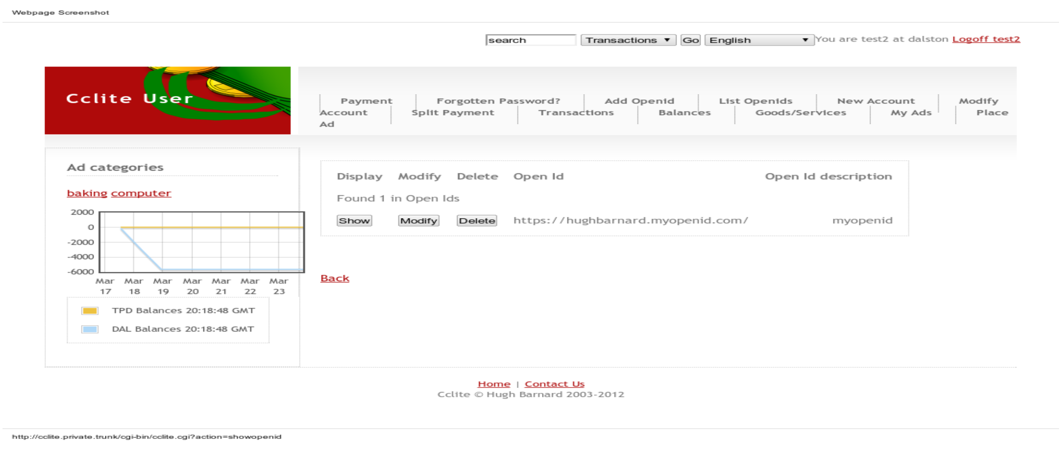
Adding An OpenId

You need this screen to add OpenIds. The description is just anything that lets you remember them. It's probably better to only to use OpenIds that are connected with your cclite installation [for example a connected content management system or social network].



Listing, Modification and Deletion

If you list OpenIds, you can also modify them and delete them.



Display	Modify	Delete	Open Id	Open Id description
Found 1 in Open Ids				
Show	Modify	Delete	https://hughbarnard.myopenid.com/	myopenid

7. Technical Description

This section describes the organisation and contents of files. It also provides some guidance on changing things. Cclite is still under active development, some caution is needed.

Architecture

The architecture is pretty much model/view/controller. The controllers are `cclite.cgi`, `ccadmin.cgi, ccsmsgate.cgi, ccserver.cgi` (the SOAP controller) and `ccinstall.cgi`. The views are provided by the templates but this is currently imperfect (there's still quite a bit of presentation generated in `Cclite.pm`; one of the main modules, for example gradually being moved to `Ccu.pm` the utility module).

`ccsuggest.cgi` contains the logic for making automatic suggestions for user names, transaction destinations and searches. If you don't want this feature, remove the script or remove its execute permissions.

The `batch` subdirectory contains batch interfaces for rss, csv and gammu, for example. These are now cgi style programs and can be started from the admin control panel. The graph programs are being replaced by Ajax calls and flot: http://code.google.com/p/flot/ generated graphs as of 0.9.0

`Ccdirectory.pm` contains most of the user adverts and yellow pages logic. This is to enable users or groups to write their own version of this.

The 'business' logic is provided in the `Ccxxxx.pm` series modules.

The database code and SQL is in `Cclitedb.pm` and therefore, in principle, a new database or file store can be implemented without impacting the rest. Some specific SQL is provided as-is using `sqlraw`. This is gradually being converted into specific routines.

`Cchooks.pm` is included and will contain a programming interface for the cautious, routines that allow pre and post transaction code without modifying the main code base, for example. We haven't done very much with this to date.

Subdirectory Layout

This is a summary description of software contents. Note, the major subdirectories are mentioned because the main directory depends on the install path. For example `cgi-bin` is probably `/usr/share/cclite/cgi-bin` and `public_html` is `/var/www/cclite/public_html` in a Debian or Ubuntu style installation.

Subdirectory	**Contents**	**Notes**

`cgi-bin`	`cclite.cgi`
	`ccserver.cgi`
	`ccsuggest.cgi`
	`ccopenid.cgi`
		Main controller for cclite
		Server for SOAP web services
		auto-suggest for main scripts
		openid login processing

`cgi-bin/protected`	`ccadmin,cgi`
	`ccinstall.cgi`
	`ccsmsgate.cgi`
		Administration controller
		New registry creation. `ccinstall.cgi` should be removed or disabled when the install is finished

`cgi-bin/protected/graphs`	`graph.pl`	Graphing using GD and GD::Graph

`cgi-bin/protected/batch`	`readcsv,pl`
	`readmail.pl`
	`read_from_jabber.pl`
	`read_pop_mail.pl`
	`read_pop_mail_gpg.pl`
	`writerss.cgi`
	`readsms_from_gammu.pl`
		These scripts have been moved so that they can be run from the web rather than being cron only. However they will require work for running under cron

`lib`	Cclite Perl library modules also including `Simple::Template`	Controllers need to find this directory, it's hardcoded as relative in the controller scripts as of 0.7.0

`literals`	`literals.en`
	`literals.fr`
	`om_categories.csv`
		System messages and a csv file for small ad categories to permit modification and translation

`doc`		Manual and other documentation. Includes html pages documenting the program code in doc/html
`sql`	Registry creation template	Contains SQL for a registry database including manager login and a sysaccount. sql snippets for upgrading previous versions.
`template/html/en`	English language templates	Templates can be added in template/<language code> for example

`public_html`	Intial setup page:
	`index.html`
	`.htaccess`
		`index.html` needs to be removed after setup `.htaccess` supplies rewrite rules for REST

`public_html/javascript`		All javascript libraries. Mainly `jquery` now but there's quite a lot of custom code in `cclite.js` which include some language dependent messages
`public_html/images`		Images and the graphs are written into /images/charts/<registryname>/
`public_html/out`		processed batch transaction results and rss are written here into subdirectories,. May be worth protecting this with `.htaccess`
`php`	`curlclient.php`	Experimental curl based client for the REST interface
`gateway`		Contains experimental example gateways for Elgg and Drupal

`config`	`cclite.cf` `readjabber.cf` `readmail.cf` `readmailgpg.cf` `readsms.cf` `logging.cf` `gammu.cf`	Used for active configuration files. These are likely to be merged into a Windows [boo] style sectioned configuration sometime real-soon-now.

Customisation

This section deals with other languages, adding, removing and changing things.

Improving Languages and translating

There are now very rough, machine translated templates for about 15 languages delivered . There are also machine translated literal files and all the javascript messages are loaded from the literals file, so those are multilingual as well. It isn't perfectly multi-lingual ready yet. I'm not sure what would happen with multi-byte languages, either, however the rough Chinese script looks pretty good, even if Sook says that some of it is strange [*hydraulic ram* and *water sheep*: http://www.electriceditors.net/edline/vol6/6-12.txt for those who know that story].

Cclite uses `Simple::Template` for its screens and thus can be translated at about 90% without great difficulty.

The second part of the language specific items are in `literals/literals.<language-code>`. These are also roughly translated, for example `literals.fr` is the French version. There is still some html in the Perl modules, this is gradually being isolated in Ccu.pm.

Finally the small ad categories that are in the `om_categories` database table would need translation. However, with version 0.9.0 there is free form tagging rather than fixed categories. This needs a little development still.

To translate screens; copy out the English ones into another subdirectory in templates, for example `templates/html/hu` for Hungarian and translate. Then, add an extra drop down item to the user preferences part and to the main menu. Please use: http://en.wikipedia.org/wiki/ISO_639-1 codes!

If you do this, please tell me via the google group http://groups.google.co.uk/group/cclite or my contact form and I will make them available generally, if possible.

Changing current screens

The screens are style sheet controlled via `styles/default.css.` This is the best method for changing the appearance of things (colours, font sizes etc.) globally. When changing individual screens, beware of the template tags, for example

```
$$fieldsref{name}
```

`!$$fieldsref{name}` *this one is the name of a file must be on its own line as first item*

If you want to make major modifications to the screens, read the `Simple::Template` documentation at http://www.cpan.org

Adding and Removing Functions

If you want to remove a function, remove it from the appropriate screen AND comment it out in the appropriate controller (otherwise it is still reachable via URL hacking, for example!). There's more about this in the general programming section too.

Also, if you don't want to install any more registries or make currencies, remove `ccinstall.cgi` after using it, for example. It's a *very good idea* to off-line, take away execute permissions or remove it between modifications anyway.

If you want to add a function:

- Specify the function (always a good idea!)
- Add the call to the function in the appropriate controller
- Code and test the function (which is usually added to `Cclite.pm`, the main logic part or `Ccadmin.pm` if it's an administration function
- Contact me, if you want, to put the new function in the main distribution *but remember my code in Cclite has a GPL licence .*

See also, the explanation of `Cchooks.pm` for adding pre-processing and post-processing to transactions. This is currently rather underdeveloped.

Adding and Removing Batch Operations

Preferably, use one of the existing batch operations as a model. A new batch operation requires a new script in the `batch` subdirectory and perhaps a new `cron` job.

If it's some type of transaction processing; it should provide data to `sub transaction` in `Cclite.pm`. This means that there's a unified interface for all transaction operations (and that anything in `Cchooks.pm` will also be called).

Implementing Demurrage and Commissions

For example, intuitively, demurrage operations might be a once-a-month batch operation on accounts that are defined as inactive (within the money system that is implemented, this will vary from place to place). Commissions to a central account could also be implemented in this way.

You can also use and amend the processing that does the service charge.

Changing directory layout

I don’t really recommend this. Mail me and tell me why you want to do it and I’ll have a think about the reasoning though.

For example, I previously had the templates within the web root. But they shouldn't be accessed as web pages (since they are not complete, pages themselves), so I moved them 'above' the root.

The new base layout is Debian (and therefore Ubuntu) oriented because that is a popular distribution as of late 2008. However the build script now generates layouts for linux-generic/webmin style subdomains, Cpanel style and Windows XP (I'm not bothering with Vista, may look at Windows 7/8 if someone lends me a system or tries it out, I'll help).

Cchooks.pm Specialised transactions

This is under development, currently. It’s a somewhat empty module where cautious coders can put pre and post processing code for new transactions and users etc. it should have all input data available for the given type of operation.

It will enable transaction commissions for example.

Adding and Modifying REST Operations

To do this, add or modify the rewrite rules within: `public_html/.htaccess`
The security level for the REST interface is medium at present, certain operatons are only allowed with a merchant key and from a specific IP address, given in the registry record. No doubt there could be more still,.

There's a rough connector for Drupal, for Elgg and a simple example Php Curl client in the gateways subdirectory. *None of these are production quality*, but they show the kind of thing that's possible.

SOAP based customisation

Theoretically, most of `Cclite.pm` (for example) can be accessed remotely from other pieces of software, written in other languages. It was certainly my view that `Cclite` should be able to exchange data with other software based registries.

I also see that it may be useful to build and allow federated searches for goods and services within a region, for example. SMS payments also have a standard title field at present, The goods and services aspect can also be catered for, to some extent, by federating `rss` feeds from various registries in the region. As far as I'm concerned, convenience, utility and adoption are more useful than theory; so I'm waiting and seeing.

The `SOAP` accessible and federated aspects (of which payment is the most obvious and the original one) can also be used to build geography based displays for items, trading health and schemes. I may try and build a demonstration based on the `Google Maps API` or better Open Street Maps.

Finally, SOAP based federation from a single point (giving star shaped graphs) may not be optimal. I've been thinking about agents that could visit accessible registries and cumulate offers and wants, for example. Some vague intuitions about graph theory (especially the concept of a Hamiltonian, see http://en.wikipedia.org/wiki/Hamiltonian_path, for example) suggest that this might be more efficient. The computer science would be fun as well, an important point.

8. Debian Package Structure and Setup

This is an beta-version package and therefore, be careful! Preferably, install it first of all on a test machine. There are instructions for forced removal of the package later on. Normally, the package should remove with `sudo apt-get remove cclite`.

You still need to read rest of the manual as well! This simply makes installation easier.

The included virtual server definition gives aliases to run the scripts out of `/usr/share/cclite/cgi-bin` and gives a web root under `/var/www/cclite`. You can use this as a starting point for something more evolved. All the previous documents are in `/usr/share/doc/cclite`. I've tried to follow the Debian policy manual and the file system hierarchy as much as possibly but this is version 0 of the package. Your mileage may vary.

Package Structure

The payload for the package (which can be used for a non-automatic tar-style installation too) are organised as follows:

`./usr/share/cclite`	Contains the motor, scripts and libraries. The batch scripts can now be run from the admin part of the cgi and are in cgi-bin/protected/batch and cgi-bin/protected/graphs, including one for starting gammu
`./var/www/cclite/`	Contains the part that is visible as the web root. Also directories that can be written by rdf files and processed batch files
`./var/cclite`	Contains directories and subdirectories that are used by the batch processes. For example /var/cclite/sms is used by gammu and the batch readsms_from_gammu.pl
`./etc/apache2/sites-enabled`	Contains a virtual server definition that will enable a Debian machine to be used as a private stand-alone on an intranet or used for experimentation by an individual.
`./etc/hosts`	Contains a reference to the virtual server host name, for use with the above.

Non-package installation

Move all these directories and files into their standard positions under the root (`./var/cclite` goes to `/var/cclite` for example) and restart or start Apache. Then go to http://cclite.private.server/cclite where there is an index page with requirements checker and installation links. The install (except for SMS described later) is as previous releases.

Package Installation

Download the package and double-click on it. If the requirements (Apache,. Mysql and various Perl modules) are satisfied, the package will install. Install scripts should restart Apache as well.

It's recommended to **pre-install the** mysql-server package as there's some difficulty installing it as an automatic dependency package (input root password, for one). Also I had to pre-install `libpq5` (`sudo apt-get install libpq5`) on one test system.

Then go to `http://cclite.private.server/cclite` where there is an index page with requirements checker (new) and installation links. The install (except for SMS described later) is as previous releases.

Emergency Package Removal

As this is a preliminary package, it may prove problematic, so you can force its removal with: `sudo dpkg --force-all -r cclite` followed by `sudo dpkg --purge cclite` to clean up files.

9. RPM Package and Setup

This is an alpha-version package and therefore, be careful! Preferably, install it first of all on a test machine. There are instructions for forced removal of the package later on. Normally, the package should remove with `sudo rpm -e cclite`.

You still need to read rest of the manual as well! This simply makes installation easier.

The package has been tested on 64 bit Redhat instances on Amazon EC2. Indeed, nearly all package testing both Debian based and Redhat is EC2, as this means not running, clearing down physical machines in our own server rooms. However, this doesn't offer guarantees for other versions and specific releases of Linux.

Package Setup

This is an extract from the spec file for the rpm package. The file system layout is as for Debian, in `/usr/share/cclite` and `/var/www`. The post-install is also pretty-much the same as Debian, the idea being to maintain two very similar packages that respect the file system hierarchy for Linux [and cut down on the build and maintenance work, laziness of of the Perl virtues!].

```
Summary: Alternative Currency Software Cclite
Name: cclite
Version: 1.0.1
Release: 1
License: GPL+
Group: Applications/Productivity
BuildArch: x86_64

%description
Alternative currency software for LETS and other currency systems
Can handle multiple currencies.

 # Provide perl-specific find-{provides,requires}.
 %define __find_provides /usr/lib/rpm/find-provides.perl
 %define __find_requires /usr/lib/rpm/find-requires.perl

%files
/usr/share/cclite/
/var/www/cclite/
/etc/httpd/conf.d
%post -p /usr/bin/perl
```

```
# based on debian postinstall
use strict;
my $operation = $ARGV[0];
$| = 1;
my $web_root    = '/var/www';
my $cclite_root = "$web_root/cclite";

# 1 is a new install, only operation supported at present
if ( $operation == 1 ) {
    # make configuration file writable by apache
    # note this is www-data for debian
    `chown -R apache /usr/share/cclite/`;
    `chmod -R u+w /usr/share/cclite/config/`;
    # make var/cclite owned by apache, gammu must be in this group therefore
    `chown -R apache /var/cclite/`;
    `chmod -R u+w /var/cclite/`;

    # make the output web root directory writable by apache + group
    `chown -R apache /var/www/cclite/`;
    `chmod -R u+w /var/www/cclite/`;

    # log file permissions for log4perl, now removed as of 07/2012, no log4perl
#     `chown   www-data /var/cclite/log/cclite.log` ;
    `chmod -R  u+w /var/cclite/log/` ;
    `/etc/init.d/httpd restart`;
    #FIXME: Doesn't deal with upgrades and reconfigs at present
}
elsif ($operation == 2) {
    print "package cannot upgrade at present\n";
    print "please remove and re-install\n" ;
    exit 1;
}
print "restarting apache, cclite is available at http://localhost now\n" ;
`service httpd restart` ;
exit 0;
%changelog
* Sat Jan 3 2014 - First version of cclite rpm
- Probably somewhat flaky
```

Package Removal

```
sudo rpm -e cclite
```

Should do it, but some manual clean up in /etc for example may be needed. As this is brand new, your mileage may vary.

Redhat Security and IP Tables

These are the commands that were necessary on EC2 to support the package. Also the setenforce and iptables commands turn a great deal of the internal security off. You may not with to do this and therefore you will need to tailor your own solution.

```
# rough outline for installing cclite on amazon redhat
yum install httpd mysql-server mysql perl gcc

sudo /etc/init.d/mysqld start
# this is the password that will go in the cclite configuration
/usr/bin/mysqladmin -u root password 'mysql-password'

sudo service httpd start
sudo setenforce Permissive
sudo /etc/init.d/iptables save
sudo /etc/init.d/iptables stop

sudo rpm -U -vv -h --nodeps cclite-0.9.3.x86_64.rpm
```

10. SMS Architecture and Setup

There are two ways of processing SMS transactions (confirmations, joins, suspends, PIN changes, payments and balance enquiries):

– Local gateway with mobile phone or GSM dongle
– Commercial gateway providing web transactions to `ccsmsgate.cgi`

Local Gateway

The local gateway which uses Gammu.pm and readsms_from_gammu.pl is considerably improved and tested for Cclite 0.9.3. For example, a user can join a trading group from a mobile phone. The intention is to make this part, pretty much stand-alone, for people that use a phone and don't use the web very much. We've stayed with SMS, because people don't necessarily have smart-phones and, also, 'apps' [which we 'may' do for convenience] are a much more closed architecture. This is described a little later,

Commercial HTTP Gateway

`ccsmsgate.cgi` was written to connection with two specific providers in the UK and therefore it's likely that it will need modification for alternative suppliers. The processing for suppliers and Gammu is provided via `lib/Ccsms/Aql.pm`, `lib/Ccsms/Cardboardfish.pm` and `lib/Ccsms/Gammu.pm`. So you need to use the specific library.

Also `lib/Ccsms/Cardboardfish.pm` is the most evolved and will provide SMS receipts [*these can be switched on and off at user level, since they have a national currency cost*] and charges the user 1 sms unit for the receipt. *It's probably the most useful model for building further supplier interfaces.*

The principle (gateway processes SMS and supplies the result as a web form post) is usually similar for most of them, though. In all cases, at the moment, the type of processing [Aql, Carboardfish or Gammu], the registry and currency are set up via `readsms.cf` which is read in `ccsmsgate.cgi` and the `lib/Ccsms` module, for example:

```
readconfiguration('/usr/share/cclite/config/readsms.cf');
```

You may need to change the path for this when setting up and will need to edit the file.

SMS Emulation Page

There's `../public_html/html/en/smsemulation.html` page used to test message formats and modifications without using a gateway or gammu. This is not intended for live use and will need modification for your specific message format. It's there as an example of how to test without 'wasting' loads of SMS messages. It's probably better to remove this page or make it unreadable, if you're not using it currently,

Local Gateway with Mobile Phone

It's now been tested with a Huawei 1750 model GSM Dongle. However, some work needs to be done to attach multiple mobile phones and therefore deal with heavier traffic.

Gammu from `gammu.org` is now used as a server for receiving SMSes. Thus, a much wider range of phones, including some older ones can be used. Also, because it's part of a big project gets properly maintained.

Note that the component that reads the SMS messages is now `gammu-smsd` the sms daemon. This has changed in a recent gammu release.

The new script `readsms_from_gammu.pl` just reads the SMSes that it finds in `/var/cclite/sms/inbox` or whichever `smsinpath` is configured in `readsms.cf` and processes them, either locally or on another system using SOAP as a transport (We'll probably move the SOAP to REST, in a while).

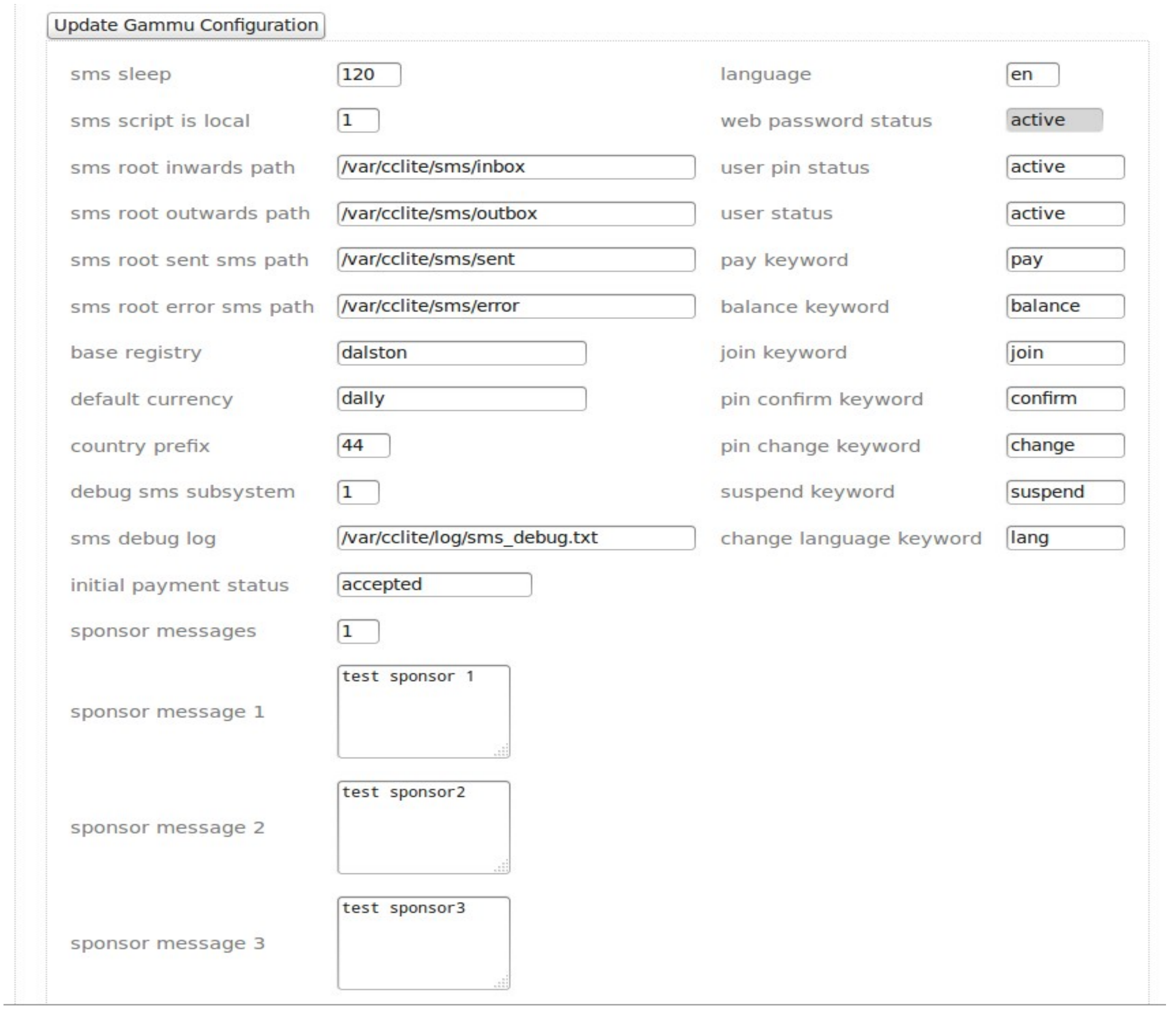
Message Directory Structure

`/var/cclite/sms`	This is the toot for SMS files
`inbox`	Messages received to be processed
`outbox`	Messages to be sent to users, mainly acknowledgements
`sent`	Sent Messages
`error`	Messages in error

For each of these, there is also a registry name, so that the set-up will permit multi-registry operation. For example, the inbox for dalston would be: `/var/cclite/sms/inbox/dalston`

Web Based Configuration

`readsms.cf` now has a web based configuration page for Gammu. This is shown below.



sms sleep	120	language	en
sms script is local	1	web password status	active
sms root inwards path	/var/cclite/sms/inbox	user pin status	active
sms root outwards path	/var/cclite/sms/outbox	user status	active
sms root sent sms path	/var/cclite/sms/sent	pay keyword	pay
sms root error sms path	/var/cclite/sms/error	balance keyword	balance
base registry	dalston	join keyword	join
default currency	dally	pin confirm keyword	confirm
country prefix	44	pin change keyword	change
debug sms subsystem	1	suspend keyword	suspend
sms debug log	/var/cclite/log/sms_debug.txt	change language keyword	lang
initial payment status	accepted		
sponsor messages	1		
sponsor message 1	test sponsor 1		
sponsor message 2	test sponsor2		
sponsor message 3	test sponsor3		

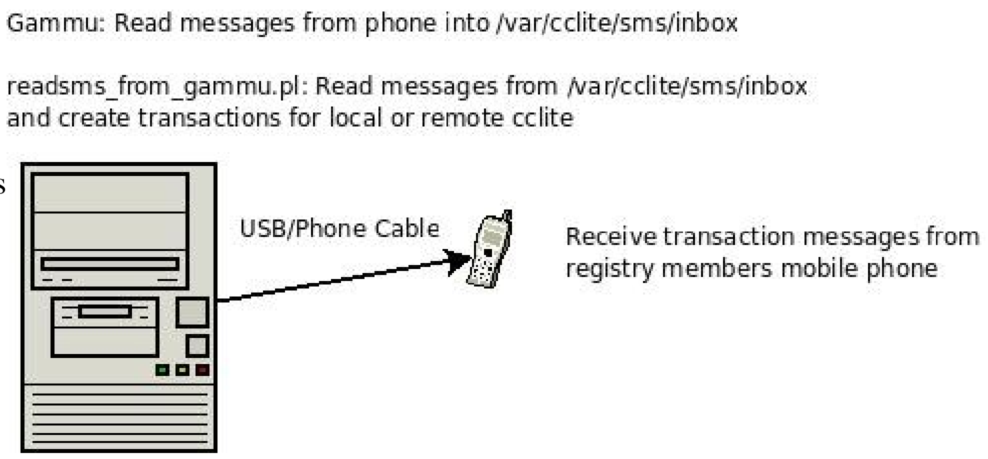
Parameter Descriptions

Parameter Name	**Description**
`sms_sleep`	Sleep in seconds between trying to read messages
`sms script is local`	1 if the script is on the same system as the rest of cclite. Otherwise this needs SOAP to access the Clite libraries, this has not been extensively tested
`sms root inwards path`	As described, inwards directory for SMS
`sms root outwards path`	As described, outwards directory for SMS
`sms root sent path`	As described, sent directory for SMS
`sms root error path`	As described, error directory for SMS
`base registry`	Registry that will be default if not named in SMS. It's usual for the SMS system to be one registry
`default currency`	Curreny that will be default if not named in SMS. It's usual for the SMS system to be one currency
`country prefix`	Default dialling prefix (44, 33 etc.) if not mentioned in the SMS itself. Usually the country where the system is located.
`debug sms subsystem`	1 to switch on sms debugging. In this case SMS details are written into the log but not sent
`sms debug log`	File path for the debug log
`initial payment status`	Initial payment status, usually 'active' for SMS based systems
`sponsor messages`	Switch on sponsor messages at bottom of SMSes
`sponsor message 1 - 5`	Sponsor messages that are randomly rotated at the bottom of SMS messages
`language`	Default language code
`web password status`	Web password status for users that join via SMS. This needs to be 'active' otherwise they can't login

`user pin status`	PIN status for users that join via SMS. This is usually 'active' unless there's a vetting process by the administrator, for example
`user status`	User status for users that join via SMS. This needs to be 'active' otherwise they can't login
`pay keyword`	Enables use of non-english word for pay, experimental
`balance keyword`	Enables use of non-english word for balance, experimental
`join keyword`	Enables use of non-english word for join, experimental
`pin confirm keyword`	Enables use of non-english word for PIN confirm, experimental
`pin change keyword`	Enables use of non-english word for PIN change, experimental
`suspend keyword`	Enables use of non-english word for suspend, experimental
`change language keyword`	Enables use of non-english word for changing language, experimental

It doesn't need `cron` definitions and modification, if you run it from the administration page. It does need a bit of set up and variable definition in the top of the script at the moment.

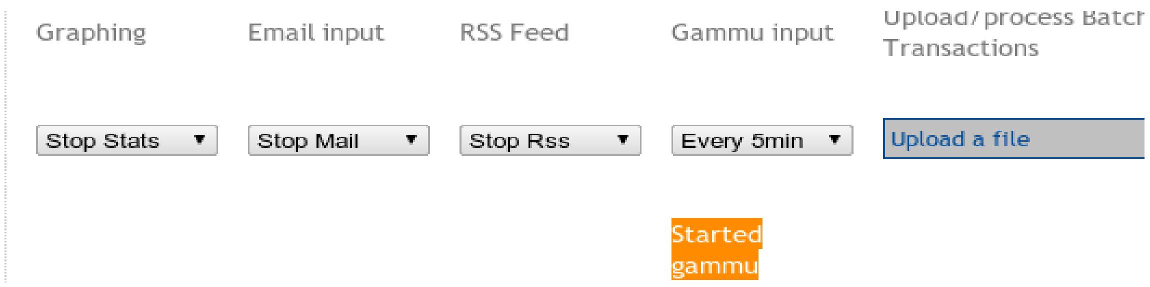
You'll need a cell phone and cable that is compatible with gammu. They are listed on the gammu website at `http;//www.gammu.org`.


Gammu: Read messages from phone into /var/cclite/sms/inbox
readsms_from_gammu.pl: Read messages from /var/cclite/sms/inbox and create transactions for local or remote cclite
USB/Phone Cable
Receive transaction messages from registry members mobile phone

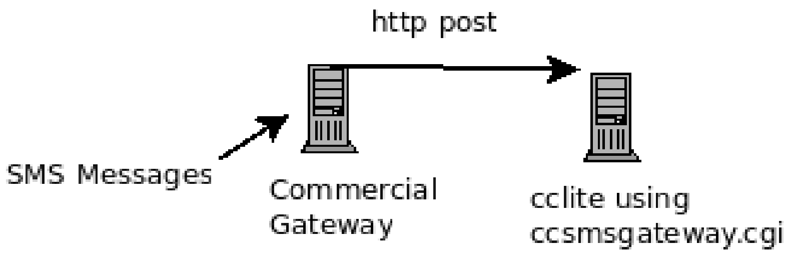
Currently, as delivered it expects a Ascii character set, however if you use encode/decode, processing UCS-2 SMSes is also possible (not much testing though, feasibility proved, mainly).

Starting Up

- Change `/usr/share/cclite/config/gammu-smdrc.example` and move to `gammu-smdrc` after updating the phone model and connection method. Wammu, the front end can be used to automatically detect this .
- To start gammu using `/usr/share/cclite/config/ gammu-smdrc` do:
  `/usr/share/cclite/batch$ sudo ./gammu.sh`
  This will be a little neater in a near future release.
- Then you can start `readsms_from_gammu.pl` from the web administration page :



Commercial Gateway



This is the architecture for use with a commercial gateway. It may involve modifications with `ccsmsgate.cgi` for processing the incoming cgi data, if you don't use Aql, Cardbaordfish or the local phone and Gammu.

SMS Message format and protocol

The target registry and currency are currently read from `readsms.cf` at the moment. The SMS messages now contain a PIN number which is set up with the account and must be confirmed by a first SMS confirm pin message. If the wrong PIN is given three times, the account is locked for SMS, It is unlocked by the manager, when the PIN is changed. Obvious PINs such as 1234, 1111 will be rejected too. The message formats are:

Operation	**Example SMS Message**
Confirm pin	`p123456 confirm`
Change pin	`p123456 change p345678`

There are many payment formats now. Cross registry is possible if the 'other' registry is local. In the example the 'dally' is an example currency.
 	
It may be useful to decide on one format, if requirements are simple. In which case, some of the parsing can be removed from Gammu.pm inside `_sms_payment_parse`
	`p2323 pay 5 to 441234567890`
	
	`p2323 pay 5 to 01234 012 345`
	
	`p2323 pay 5 to 07534 123456`
	
	`p2323 pay 5 to 01234567890`
	
	`p2323 pay 5 to +441234567890`
	
	`p2323 pay 5 to test1`
	
	`p2323 pay 5.2 dally to 01234567890`
	
	`p2323 pay 5.2 dally to test1`
	
	`p2323 pay 5 dally to 01234567890`
	
	`10 to 441234567890|test2 for numbering`
	
	`10 limes to 441234567890|test2 for numbering`
	
	`send 5 limes to test2 at dalston for demo`
	
	`10 limes to 441234567890|test2 at dalston for payment`
	
	`441234567890|test2 10 limes for payment`
	
	`441234567890|test2 10 limes`
	`# 5 to 01234 123456 for stuff`

Query Balance	`p123456 balance`
Join the default registry. This sends back a setup message with a web password and a pin for sms etc. everything is active at this stage sets up with sms receipt switched on. Information should be completed via the web.	`join hugh barnard hugh.barnard@example.com`
Suspend the account in case of fraud etc.	`p123456 suspend`
Change user language. Note that there are still some problems with fully multi-lingual SMS. Chinese doesn't work at all, at the moment.	`p123456 lang es`
Use specific configuration [experimental]. This is the equivalent of 'short codes' used by the commercial suppliers. It means that you can use a set of different configuration files within the same installation. The short code is always 'c' and a number, `c12`, `c99` etc.	For example `c1 p123456 suspend` will use `readsms-c1.cf`.

11. Web and Other Resources for Alternative Money

Remember, help with Cclite itself at: http://groups.google.co.uk/group/cclite

This list is small, partial and not in any particular order. I'd also currently recommend, *Healthy Money, Healthy Planet Developing Sustainability through New Money Systems* by *Deirdre Kent*. This is a little New Zealand centric but is a good round up of many of the problems with conventional money and reasons for doing other kinds of money:
http://www.craigpotton.co.nz/products/published/books/booksocial/healthymoneyhealthyplanet

Also the recent Thomas Greco book, the End of Money and the future of Civilization referenced on:
http://www.reinventingmoney.com/

If you want to comment about this list or lobby for something else to be included, write to me:

http://www.newciv.org/ncn/moneyteam.html
http://www.opentransact.org/
http://groups.google.com/group/agile-banking
http://www.openmoney.org/
http://thetransitioner.org/wen/tiki-index.php?page=Open+Money
http://www.letslinkuk.org/
http://ccit.wji.com/tiki-directory_browse.php?parent=36
http://www.transaction.net/
http://www.reinventingmoney.com/riegel.php
http://twenteenthcentury.com/uo/index.php/FacultyEconometrics
http://en.wikipedia.org/wiki/LETS
http://en.wikipedia.org/wiki/Money
http://www.gdrc.org/icm/lets-faq.html
http://www.reinventingmoney.com/worglCurrDemurr.php

INDEX