CXP protocol Intertrading API 1.0 (beta)

for community currencies
The Intertrading Protocol Group, October 2013

Introduction

This protocol provides a currency conversion mechanism which increases the purchasing power of complementary currencies. Using the Community Exchange Payments (CXP) protocol and API, complementary currency systems can reach agreements to trade with other communities, and their users would be able to spend the means of payment they have generated in one system to purchase products and services from other systems.

This conversion mechanism is articulated through a common base unit hosted in a common exchange server (ledger), which would process the clearing between different currencies.

This practice we call 'intertrading'.

The participating community exchange systems would collectively decide the rules governing intertrading transactions with other communities, taking decisions like: whether to manage exchange rates and balance of trade through internal policy, or whether to float the currency against their neighbours. An automated system to set the exchange rate according to the balance of trade should be available for communities to opt in to.

Definitions

User: a person supplying or procuring goods and services.
Currency: a unit of value and it's rules, eg: credit limits or prices (exchange rates).
Account: an account controlled by a User with a record of a quantity of currency.
Community: a group of Accounts denominated in a common Currency.
Intertrading Circle: a Community of Communities using a 'base' Currency.
Virtual Customer: the Community's Account in the Intertrading Circle.
Import-Export Account: the Account in the Community which is used to connect to the Intertrading Circle. The balance is always the negative of the associated Virtual Customer Account.
Governing Body: the governing body for each Community (including intertrading communities).
An association providing accounting services between community currencies should include the following elements in its constitution.

How to use this Protocol

Once there is a governing process in place between the member communities a number of design choices should be made as follows:

Template for a constitution.

1. Who can join?
An Intertrading Circle is a club. Membership is voluntary and at the discretion of the club's Governing Body. The club can have rules which the member Community agrees to abide by. Member Communities should implement the API preferrably using open-source software.

2. What is the joining process?
The joining process is determined by the communities within an Intertrading Circle or with whom a Community is intertrading. When a Community joins an Intertrading Circle, an account in the Intertrading Circle is created for the new community and an API key or login is given to them. They can be given credit limits according to any rules.

3. How to determine exchange rates?
All member Currencies are expressed as a multiple of a base Currency. Members can be responsible for declaring and maintaining their own exchange rates, or exchange rates can be automatically set to keep Intertrading balances between agreed limits. Joining Communities usually have a nominal currency value, so they may be able to declare this themselves. There may be a given moment, periodically, for Communities to redeclare the values of their Currencies.

4. What is the base unit?
Intertrading means that the base unit is a fixed amount and all Currencies are calculated in relation to it. In a free-markets such as Forex, the base unit floats against all the other Currencies.
Example base units: a basket of currencies, KWh, miligrams of gold, minutes, or dollars.

5. How are credit limits between member communities determined?
It is advised to start simple. Many ways are possible. Eg: Multiple of members, percentage of last 3 months trade volume or reputation.

6. What kind of monitoring should take place?
Monitoring needs to be supported by the Intertrading Circle software.
To identify fraud, transactions can be monitored for unusual activity. To identify Communities at risk of default, credit limits can be monitored.
To identify opportunities for enterprise, supply and demand (requires offers and wants info) can be monitored.
Other, eg: transactions can be monitored to check the distance fresh food is travelling.

7. Enforcement and sanctions
An Intertrading Governing Body can block a Virtual Customer Account to stop a community from intertrading.

8. Legal considerations
Laws and regulations differ in every country.

It should be possible for intertrading associations to come together in a super-association and implement this protocol between themselves. This is called nesting.

Technical Implementation

The clearing service could be implemented using a straightforward accounting application such as Cyclos. A prototype server and client already exist in Drupal.

The clearing service must

- host one account representing each member exchange, which mirrors an account on the exchanges own server, called the 'intertrading' account,
- check integrity with the member exchange.
- impose agreed balance limits, maximum and minimum, on each exchanges account
- keep a definitive backup record of each transaction
- produce reports

The system is not expected to run without human oversight. Specifically, it does not allow communities to create accounts and access credit automatically.

Authentication & Security

There is very little incentive or opportunity to steal in a mutual credit system because

- the credit cannot leave the circle
- every transaction leaves a record
- the credit has no value in itself

HMac is understood to be the most appropriate authentication method, because it is invulnerable to man-in-the-middle attacks.

API methods

We assume here that accounts are created and approved on the clearing server, not via the client and the API.

Handshake:
Client announces to server, refreshes key, and sends statement data, and server responds with a description of the rules + account stats.
POST: http://myserver.com/server

List exchanges (ideal for ajax):
Retrieve a list of other member systems, typically for populating a dropdown widget.
GET: http://myserver.com/server/clients

Prepare form (ideal for ajax):
Typically called before a transaction is attempted, to verify client validity and prepare client about its balance limits on the server.
GET: http://myserver.com/transactions

Try transaction:

The request should be repeated every few seconds to prevent a timeout.
PUT: http://myserver.com/transactions

Relay transaction:
Server relays transaction to second client.
PUT: http://myclient.com/transactions

Message structures
The content of each message should be a json array.
All quantities are measured in base units i.e. the unit of the server.

Handshake request:
```
{[
  'api' : FLOAT, //version of this API
  'mail' : STRING (63), //address of the site admin.
  'ticks' : FLOAT, //relative value of the currency against the internal unit
  'first_trade' : INTEGER (unixtime), //the date of the first trade in the system,
  'traders' : INTEGER, //Number of traders
  'transactions' : INTEGER, //number of transactions in the last 30 days
  'uri' : STRING, //The root path e.g. domain.org/mytown?q=
  'divisions' : mixed, //(optional) 01 means cents OR list of acceptable centiles separated by pipe|
  'visibility' : BOOLEAN, //(optional) Toggle whether all this data should be visible to the public
  'logo' : URL, //(optional)The absolute url of the logo of the site (optional)
  'lat' : FLOAT, //(optional) The latitude of the site
  'lon' : FLOAT, //(optional) The longitude of the site
}]
```

Handshake successful response:
```
{[
  'message' : STRING, //html,
  'status' : ENUM, // -1 means no account, 0 means blocked, 1 means trading
  'balance' : INTEGER, //balance of the client
  'min' : INTEGER, //minimum balance limit of the client
  'max' : INTEGER, //maximum balance limit of the client
  'count' : INTEGER, //number of intertrading transactions
  'keymatch' : BOOLEAN, //whether the stored key is the same as the given one.
]}
```

Prepare form request:
```
{[
  'volume' : FLOAT, //Last 365 days volume
  'integrity' : 1, //Boolean whether the balances add up to zero. Must be TRUE!
  'balance' : FLOAT, //balance of the client intertrading account
]}
```

Prepare form successful response data:
```
{[
  //according the limits of the sites account on the intertrading server
  //suitable for client-side validation
  //both values are > 0 since they indicate the max value of a transaction in a given direction
  'spend_limit' : FLOAT, //in base
  'earn_limit' : FLOAT
]}
```

List of clients successful response:

```
{[
  'servername.com' : 'Site Title',
  'servername2.com' : 'Site2 Title' //etc.
]}
```

Try transaction (RELAYED to http://client2.com/intertrade):

```
{[
  'payer' : varchar,
  'payer_url' : varchar,
  'payee' : varchar,
  'payee_url' : varchar,
  'dest_url' : STRING, //'servername.com',
  'src_url' : STRING, //'servername.com',
  'quantity' : FLOAT,
  'description' : TEXT, //255 chars
  'really' : boolean, //whether or not to write
  'temp_id' : varchar //random string the server knows not to try twice
]}
```

Try transaction:
response data:

```
{[
  'code' : INTEGER,
  'args' : array()
]}
```

HTTP ERROR CODES
This is an ad hoc extenstion of the http status codes
http://en.wikipedia.org/wiki/List_of_HTTP_status_codes
Every word with @ will be the key of an accompanying array containing replacements, for use in translation.

```
   4: 'Transaction would exceed limits on server.'
   5: 'Problem saving transaction on server: “@message”'
   6: 'Unknown account & failed to create a new account on intertrading server.'
   7: 'Diagnostics from server: “@message”'
   8: 'Invalid transaction field: @field: @value'
   11: 'Missing config field: @fieldname'
   12: 'Field @fieldname should be @rel 0: $val'
   13: 'Field @fieldname contains invalid characters.'
   14: 'Your Intertrading ratio (balance/volume) exceeds @num%: @balance / @volume'
   15: 'Type error in field @fieldname. Should be a @type.'
```

```
16: 'Your exchange is not permitted on this network.'
17: 'Wrong key.'
18: 'Not enough data to authenticate.'
20: 'Server failed to authenticate with remote client.'
24: 'Transaction would exceed remote client's account limits.'
25: 'Misc validation error on remote client: @message'
26: 'Server not found: @server'
27: '@message'', $args //anything else
```