UPB

RBM Documentation

RBM Application & Installation Help

HP

2012.

[Type the abstract of the document here. The abstract is typically a short summary of the contents of the document. Type the abstract of the document here. The abstract is typically a short summary of the contents of the document.]

Contents

RBM Application Documentation

1 Major Concepts

1.1 web2py

by Alexandru Lazar
alex@zencoding.org

Web2py is a full-stack framework with full support for:

• HTTP requests, HTTP responses, cookies, sessions;
• multiple protocols – HTML/XML, REST, ATOM and RSS, RTF and CSV, JSON, JSON-RPC and XML-RPC, AMF-RPC (Flash/Flex), and SOAP
• CRUD API;
• multiple authentication mechanisms and role-based access control;
• database abstraction layer (DAL) that dynamically generates SQL and runs on multiple compatible database backends;
• RAM, disk, and memcached-based caching for scalability;

• internationalization support;
• jQuery for Ajax and UI effects;
• automatic logging of errors with context.
Web2py uses the WSGI protocol, the Python-oriented protocol for communication between web server and web applications. It also provides handlers for CGI and the FastCGI protocols, and it includes the multi-threaded, SSL-enabled Rocket wsgiserver.
The main reasons why we chose web2py are:
• It implements a coherent, abstracted API for all the features we required, which essentially ensured that we could concentrate on developing the application without breaking any intended deployment environment.
• It is fully open-source, with a strong corporate backing and a well-crafted release cycle. Thus, we avoided any kind of vendor lock-in, while ensuring that we can continue supporting the application for several years.

1.2 Application structure

The directory structure of the application shows roughly the following view:

```
4.0K drwxr-xr-x 2 1000 4.0K Feb 25 20:20 cache/
4.0K drwxr-xr-x 2 1000 4.0K Apr 15 19:51 controllers/
4.0K drwxr-xr-x 2 1000 4.0K Feb 25 20:20 cron/
4.0K drwxr-xr-x 2 1000 4.0K Apr 15 19:54 databases/
36K drwxr-xr-x 2 1000 36K Apr 15 19:53 errors/
4.0K drwxr-xr-x 2 1000 4.0K Feb 25 20:20 languages/
4.0K drwxr-xr-x 2 1000 4.0K Apr 15 18:15 models/
4.0K drwxr-xr-x 3 1000 4.0K Apr 15 19:54 modules/
4.0K drwxr-xr-x 2 1000 4.0K Feb 25 20:20 private/
4.0K drwxr-xr-x 2 1000 4.0K Apr 15 19:54 sessions/
4.0K drwxr-xr-x 6 1000 4.0K Feb 25 20:20 static/
12K drwxr-xr-x 2 1000 12K Apr 15 19:14 uploads/
4.0K drwxr-xr-x 9 1000 4.0K Feb 25 20:20 views/
```

The controllers, models, modules and views directories contain the application code itself. the other directories contain static application content (user-generated or not, e.g. the uploads directory contains the uploaded offers) or various other files used internally (e.g. the errors directory contains error tickets).
The application follows a model-view-controller system. We tried to make the organization as consistent as possible and followed these rules:
1. All abstract algorithms are implemented as Python modules, independent of the rest of the application. For instance, the merit order computation algorithm is implemented in modules/merit.py, and in itself it offers an implementation that is entirely decoupled from the business or application logic. It can be (re)used in any application and does not depend on web2py – it only uses functions from the standard Python library.
2. These algorithms are only used in the controllers; in other words, the controllers use the modules in a manner that is relevant to the business logic. The dependence goes only one way – the controllers need the functions implemented as modules in order to work, but the modules don't care what calls them.
3. Everything that is presented to the user is presented through views. Furthermore, views are only populated by the controllers.
To give an example (which will be expanded in #2), consider the procedure of uploading an offer, which effectively works as follows:
1. The user accesses the Upload Index view through the Upload controller, which displays a form where he gives the file he wants uploaded.
2. The upload controller receives the file and, through web2py's API, stores it in a particular location for archival.
3. The upload controller calls a function in the Offer module, which builds an object of type Offer
4. The upload controller calls the object's is_valid() function (assuming the object could have been built – otherwise an error is returned at the previous step and an error message is displayed).
5. If the offer is valid, the upload upload controller does a few additional, internal checks (e.g. it ensures that the user who is logged in placing an offer for himself, not for someone else)

6. If everything went fine, the user is now notified of acceptance.

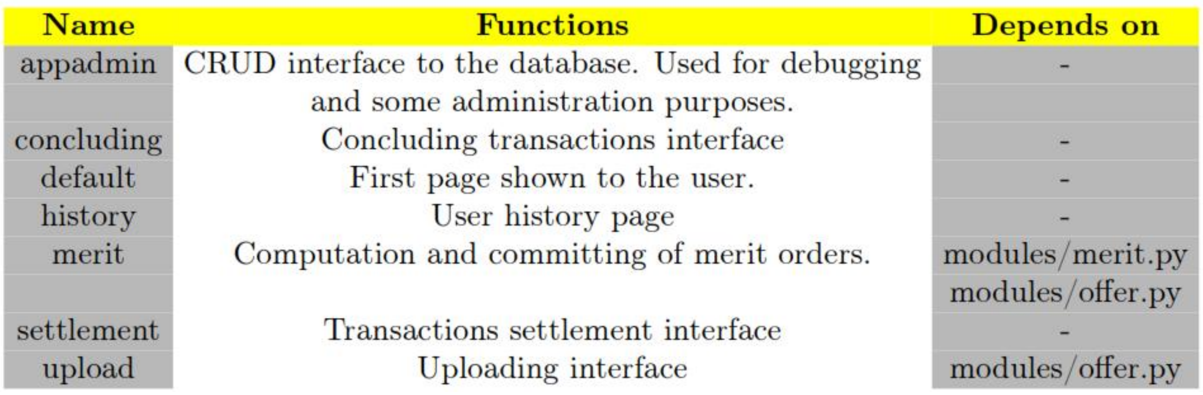
2 Application Architecture

2.1 Application Controllers and Views

There are 7 major application controllers, each with their well-defined functions.

Name Functions Depends on

appadmin CRUD interface to the database. Used for debugging -
and some administration purposes.
concluding Concluding transactions interface -
default First page shown to the user. history
User history page -
merit Computation and committing of merit orders. modules/merit.py
modules/offer.py
settlement Transactions settlement interface upload
Uploading interface modules/offer.py



Name	Functions	Depends on
appadmin	CRUD interface to the database. Used for debugging	-
	and some administration purposes.	
concluding	Concluding transactions interface	-
default	First page shown to the user.	-
history	User history page	-
merit	Computation and committing of merit orders.	modules/merit.py
		modules/offer.py
settlement	Transactions settlement interface	-
upload	Uploading interface	modules/offer.py

Table 1. Application Controllers

Each of these controllers (with the exception of the settlement controller) has one or more associated views. The views only display static content or parameters obtained from the controller, which the controller passes when calling them. For reference, the history controller looks like this:

```
{{extend 'layout.html'}}
<h1>Previously uploaded offers</h1>
{{if files == []:}}
<p>You have not uploaded any offer yet.</p>
{{else:}}
<table>
<tr align="center">
<td align="center">ID</td>
<td align="center">Ack Time</td>
<td align="center">View</td>
<td align="center">Accepted</td>
</tr>
{{for file in files:}}
<tr>
<td align="center">{{=file[0]}}</td>

<td align="center">{{=file[1]}}</td>
<td align="center"><a href="{{=file[2]}}">Click here to view</td>
<td align="center">
{{if file[3] == True:}}
Yes
{{else:}}
No
```

```
{{pass}}
</td>
</td>
{{pass}}
</table>
{{pass}}
```
Python code is always delimited by {{ and }}, and commands are executed on the server. Some code is shared by all views (which is ensured by the {{extend 'layout.html'}} directive – the base layout contains the structure of the document, the <HEAD> tags and so on). A sample output of the above view in Appendix A.

2.2 Application Database

Web2py works with a Database Abstraction Layer (DAL) which can also handle table definition and creation. Without going into much detail, the important consequences of this are:

1. There is a database model file (see below) which defines the database structure. The database is not manually created (or defined) for the particular database management system; instead, when installing the application, it detects what DBMS you are deploying it on and will generate the tables itself.
2. Inside the application, you have database introspection capability – for instance, you can check what type a field is or whether it's required or not.
3. One problem of the DAL is that it has to support a very wide range of systems, including non-relational ones. As a consequence, there are some important things that simply cannot be done portably – the most important of which is indexing. Declaring DB indices is done non-portably, through raw SQL queries, and is something that can only be handled after deployment.

web2py doesn't hold locks on the DB tables, so they can be freely shared with other applications. However, note that it does have its own interface to caching systems, so there is potential for breaking migrations.
The definition of the database module is in Appendix B. It should be fairly straightforward to read, even if you don't know Python.

2.3 Message format

The user exchanges information with the RBM system using XML files (or XML-formatted messages sent over a web service – the distinction isn't really important since what eventually gets handled to the application logic is an XML-formatted stream). An example of an XML-formatted offer is this:

```
<EnergyOfferMessage DtdVersion="2" DtdRelease="3">
<MessageVersion v="1"/>
<SenderIdentification codingScheme="A01" v="01011969"/>
<ReceiverIdentification codingScheme="A01" v="GLOBAL_RECEIVER"/>
<MessageDateTime v="2012-01-01T23:47:48Z"/>
<OfferDate v="2012-05-05" />
<Resolution v="PT1H"/>
<EnergyType v="tertiary_downward" />
<OfferType v="buy" />
<EnergyOffer>
<Interval v="1">
<Block>
<Pos v="1"/>
<Price v="51"/>
<Qty v="11"/>
</Block>
<Block>
<Pos v="2"/>
<Price v="40"/>
<Qty v="19"/>
</Block>
<Block>
<Pos v="3"/>
<Price v="33"/>
<Qty v="17"/>
</Block>
</Interval>
</EnergyOffer>
```

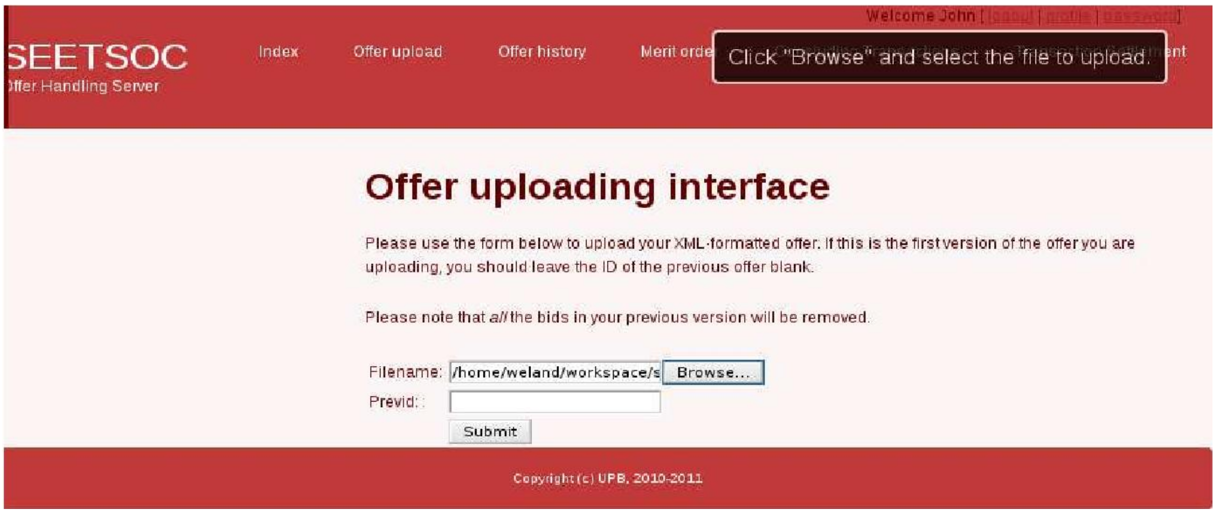
‹/EnergyOfferMessage›
These files not intended to be written by hand – they're usually generated by automatic tools. They are parsed in the Offer module, which uses a simple XML parser. It cannot do anything too advanced, but it is enough to obtain the values and validate them.
A note on representation: inside, we only use integer arithmetic to ensure we don't run into propagated errors from IEEE floating-point arithmetic. This isn't carved in stone – I haven't received any neggative feedback about the results, but if it turns out we need to use a rational representation class, it can be done easily. The representation is abstracted – there are the rtoi and itor functions defined in the repr model, which translate between the internally-used format and the external representation. If anything needs to be changed internally, modifying the two functions is all that is required to make everything work like before.

3 Example of use

The following is an example of use; note that the authorization system is disabled until we figure how we do the integration, so all functions are now accessible to all users. This is only for demo purposes; the deployed application does not allow administrators to upload offers, nor does it allow regular users to do merit order computation.

3.1 User-level usage

1. Start with an XML offer file (I have provided a set of samples). Go to the Upload Offer page and select it for uploading.


Welcome John [logout | profile | password]
SEETSOC
ffer Handling Server
Index
Offer upload
Offer history
Merit orde
nt
Click "Browse" and select the file to upload.
Browse...
Submit

Offer uploading interface

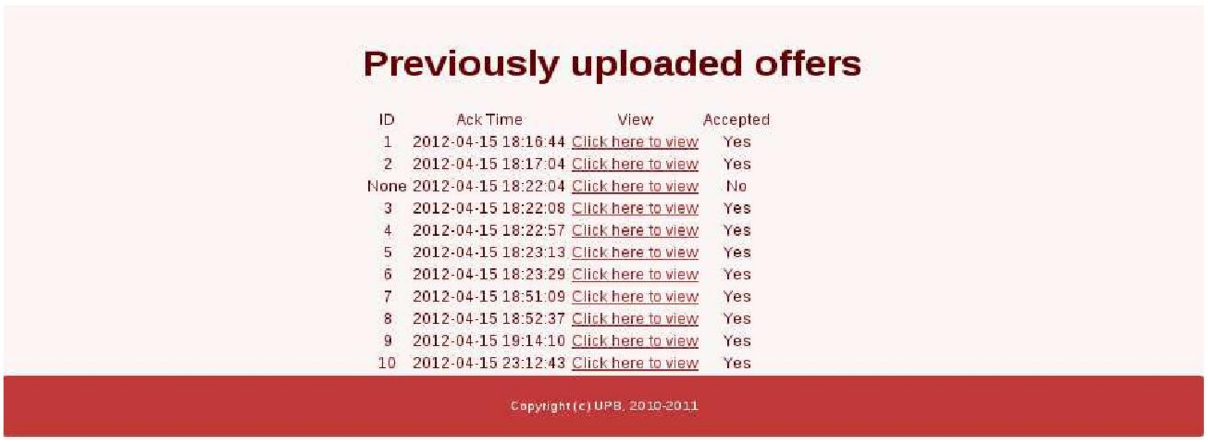
Please use the form below to upload your XML-formatted offer. If this is the first version of the offer you are uploading, you should leave the ID of the previous offer blank.

Please note that *all* the bids in your previous version will be removed.

Filename: /home/weland/workspace/s
Previd: :

Copyright (c) UPB, 2010-2011

2. Repeat (add a couple of offers to ensure that there is a representative data set).
3. At any point in time, you can view all your history in the "Offer history" page. Please note that some of the offers you see there may have actually been removed from the database. This is because we only keep the latest version. However, *all* the XML files, even the ones that have been rejected, are kept for reference and can be viewed at any time.

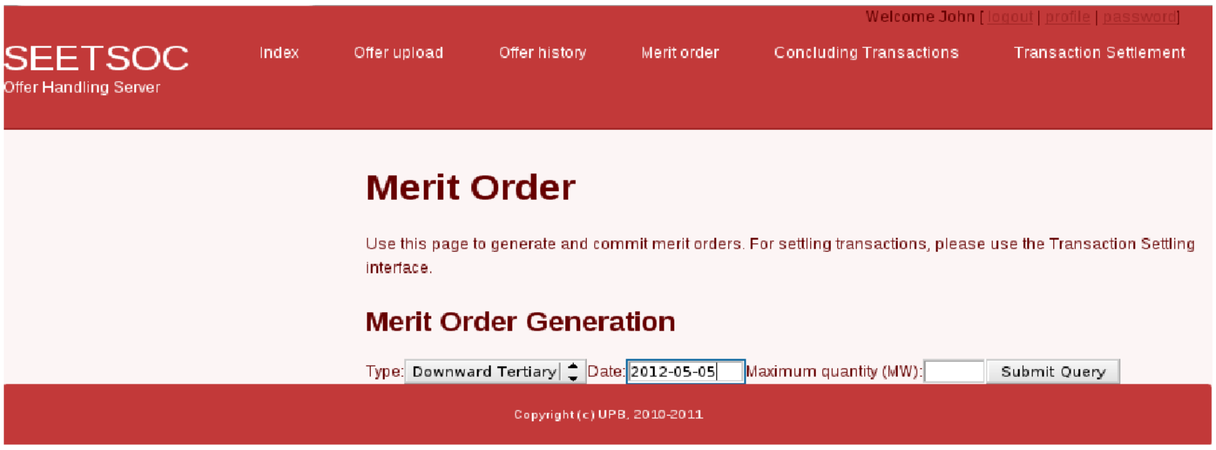

Previously uploaded offers

ID	Ack Time	View	Accepted
1	2012-04-15 18:16:44	Click here to view	Yes
2	2012-04-15 18:17:04	Click here to view	Yes
None	2012-04-15 18:22:04	Click here to view	No
3	2012-04-15 18:22:08	Click here to view	Yes
4	2012-04-15 18:22:57	Click here to view	Yes
5	2012-04-15 18:23:13	Click here to view	Yes
6	2012-04-15 18:23:29	Click here to view	Yes
7	2012-04-15 18:51:09	Click here to view	Yes
8	2012-04-15 18:52:37	Click here to view	Yes
9	2012-04-15 19:14:10	Click here to view	Yes
10	2012-04-15 23:12:43	Click here to view	Yes

Copyright (c) UPB, 2010-2011

3.2 Administration-level usage

1. Now that all offers were uploaded, let's perform a merit order computation. Go to the Merit Order page, select the date of the offers (I set them for the 5th of May 2012 for ease of use).



Merit Order

Use this page to generate and commit merit orders. For settling transactions, please use the Transaction Settling interface.

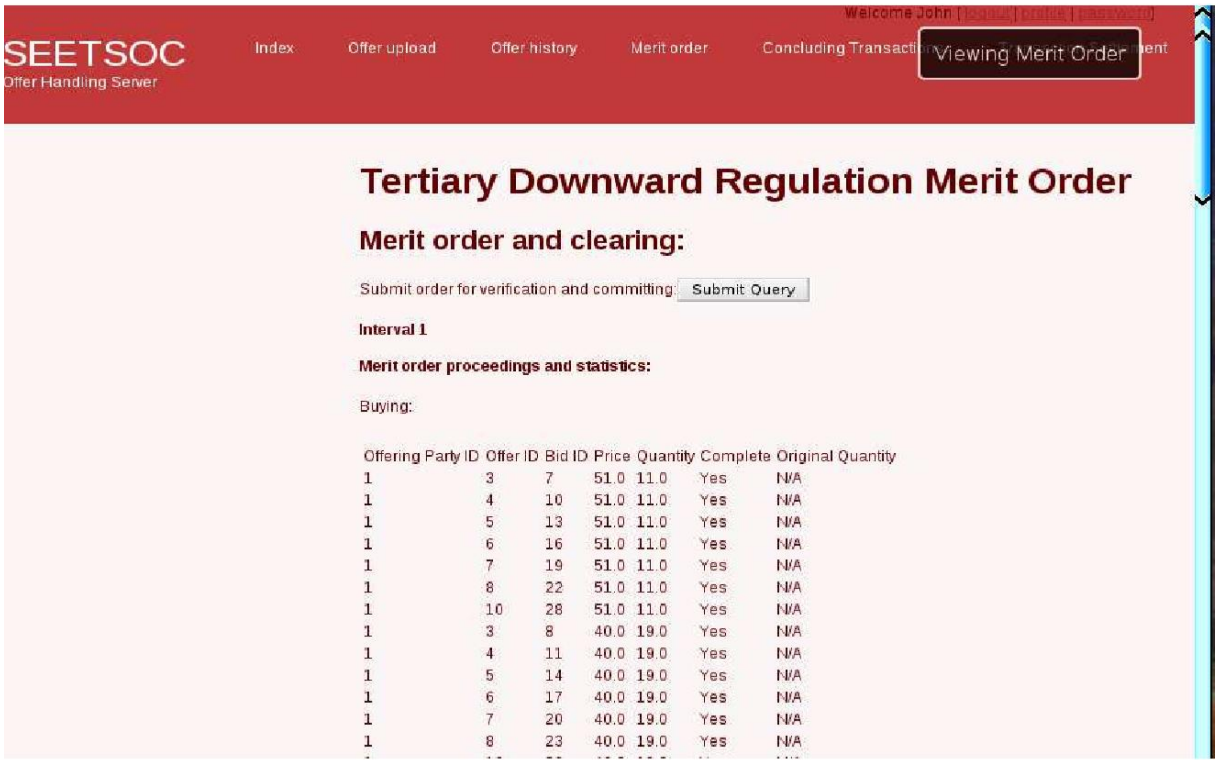
Merit Order Generation

Type: Downward Tertiary Date: 2012-05-05 Maximum quantity (MW): Submit Query

Copyright (c) UPB, 2010-2011

Example of use

2. A page is now displayed with statistics. If everything looks fine, you can commit it to the database.



Tertiary Downward Regulation Merit Order

Merit order and clearing:

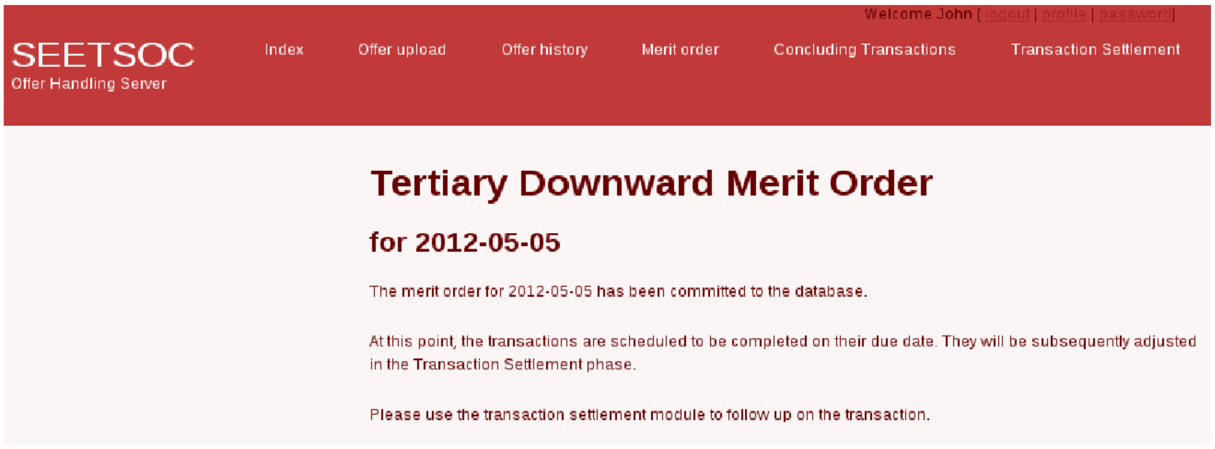
Submit order for verification and committing:

Interval 1

Merit order proceedings and statistics:

Buying:

Offering Party ID	Offer ID	Bid ID	Price	Quantity	Complete	Original Quantity
1	3	7	51.0	11.0	Yes	N/A
1	4	10	51.0	11.0	Yes	N/A
1	5	13	51.0	11.0	Yes	N/A
1	6	16	51.0	11.0	Yes	N/A
1	7	19	51.0	11.0	Yes	N/A
1	8	22	51.0	11.0	Yes	N/A
1	10	28	51.0	11.0	Yes	N/A
1	3	8	40.0	19.0	Yes	N/A
1	4	11	40.0	19.0	Yes	N/A
1	5	14	40.0	19.0	Yes	N/A
1	6	17	40.0	19.0	Yes	N/A
1	7	20	40.0	19.0	Yes	N/A
1	8	23	40.0	19.0	Yes	N/A



Tertiary Downward Merit Order

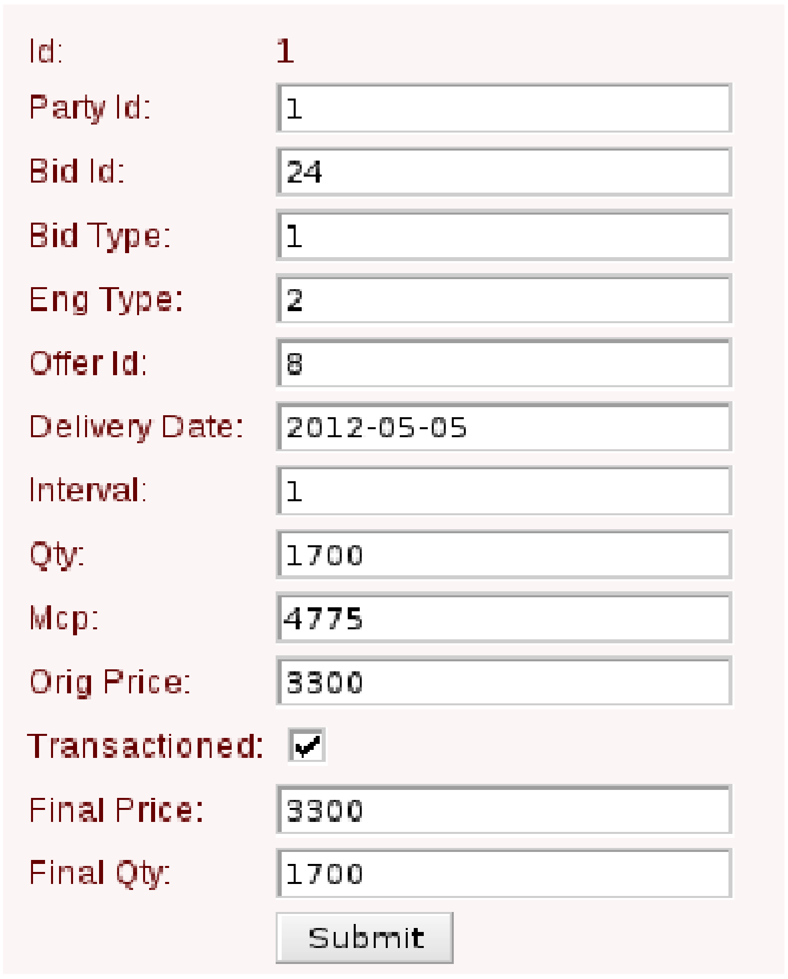
for 2012-05-05

The merit order for 2012-05-05 has been committed to the database.

At this point, the transactions are scheduled to be completed on their due date. They will be subsequently adjusted in the Transaction Settlement phase.

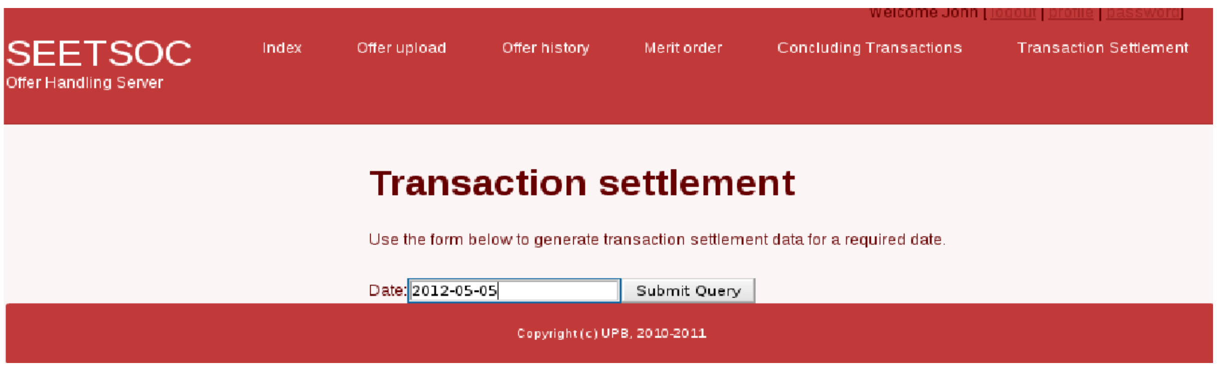
Please use the transaction settlement module to follow up on the transaction.

3. Transaction conclusion can be done from the homonymous page. Each offer is marked as transactioned as it is completed, and you can change all of their parameters at will.



Id:	1
Party Id:	1
Bid Id:	24
Bid Type:	1
Eng Type:	2
Offer Id:	8
Delivery Date:	2012-05-05
Interval:	1
Qty:	1700
Mcp:	4775
Orig Price:	3300
Transactioned:	✓
Final Price:	3300
Final Qty:	1700

4. At the end of transactions, you can generate the required transaction settlement notes from the Transaction Settlement page.



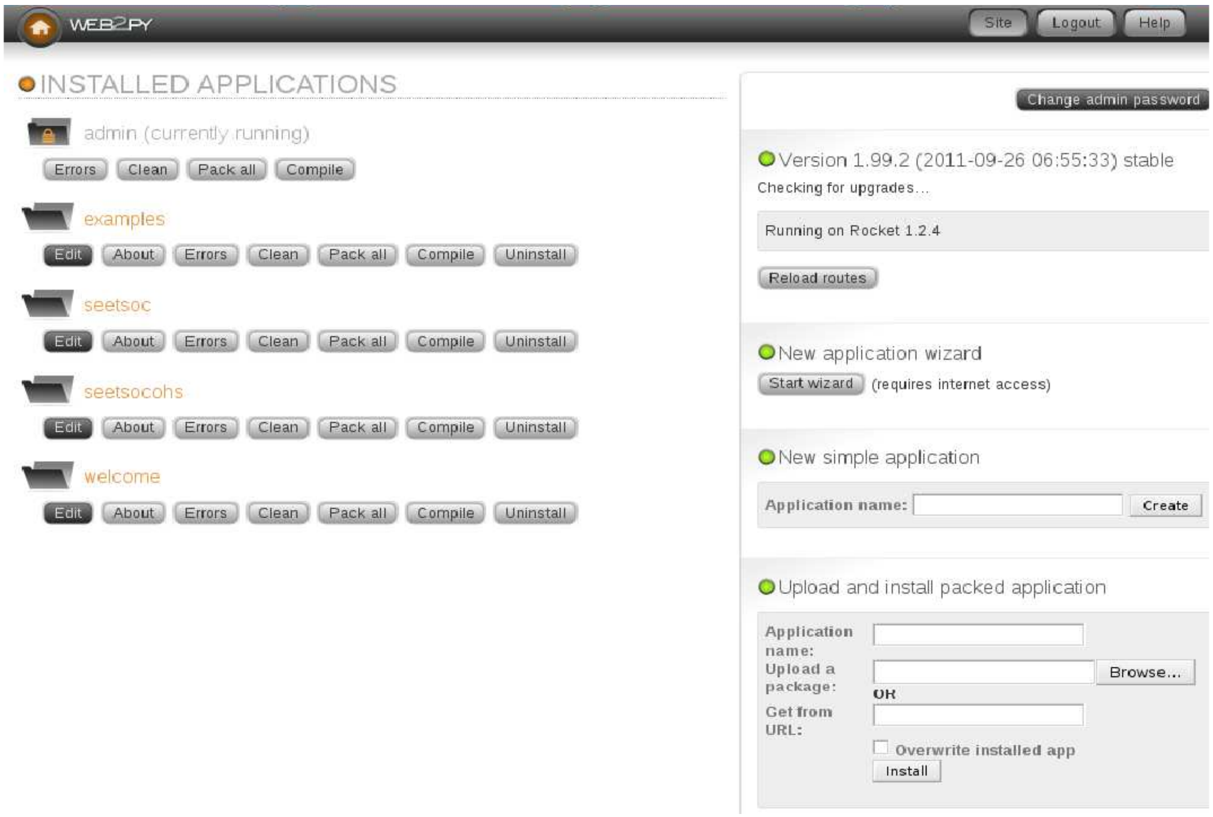
Transaction settlement

Use the form below to generate transaction settlement data for a required date.

Date: 2012-05-05

4 Installing the application

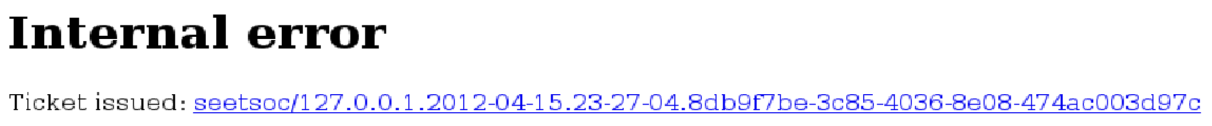
Along with this document you should have received a .w2p file. This also contains the required initial data (e.g. a set of users and TSOs to play with). Installation is done from the web2py administration console.


WEB2PY
Site
Logout
Help
INSTALLED APPLICATIONS
admin (currently running)
Errors Clean Pack all Compile
examples
Edit About Errors Clean Pack all Compile Uninstall
seetsoc
Edit About Errors Clean Pack all Compile Uninstall
seetsocohs
Edit About Errors Clean Pack all Compile Uninstall
welcome
Edit About Errors Clean Pack all Compile Uninstall
Change admin password
Version 1.99.2 (2011-09-26 06:55:33) stable
Checking for upgrades...
Running on Rocket 1.2.4
Reload routes
New application wizard
Start wizard (requires internet access)
New simple application
Application name:
Create
Upload and install packed application
Application name:
Upload a package:
Browse...
OR
Get from URL:
Overwrite installed app
Install

In the lower-right corner, find the “Upload and install packed application” box. Give it a meaningful name and click Browse near “Upload a package”, pointing it to the .w2p file. Click Install and web2py should confirm installation. That should be all. You can access it from the web2py root path with its name (e.g. if the app name is “seetsoc’, the web server is at 127.0.0.1:8000 and web2py’s root is at /, you can access it at http://127.0.0.1/seetsoc/).

5 Error tickets

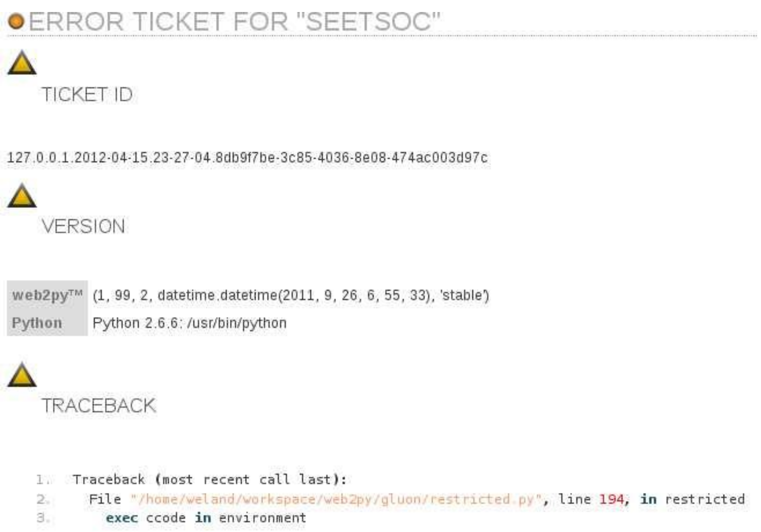
Whenever you do something Really Bad that raises an exception, you will get a page like the one below:



Internal error

Ticket issued: seetsoc/127.0.0.1.2012-04-15.23-27-04.8db9f7be-3c85-4036-8e08-474ac003d97c

I hope you won’t see it too often, but if you do, clicking the hyperlink will take you to the error ticket page.


ERROR TICKET FOR "SEETSOC"

TICKET ID

127.0.0.1.2012-04-15.23-27-04.8db9f7be-3c85-4036-8e08-474ac003d97c

VERSION

web2py™	(1, 99, 2, datetime.datetime(2011, 9, 26, 6, 55, 33), 'stable')
Python	Python 2.6.6: /usr/bin/python

TRACEBACK

```
Traceback (most recent call last):
  File "/home/weland/workspace/web2py/gluon/restricted.py", line 194, in restricted
    exec ccode in environment
```

Every time something like this happens, the application will save all its execution context, version information and a full traceback, and place them under a particular ticket ID.
You can clean up this information (or remove gathering it entirely) in production; for testing, I suggest keeping it. The main advantage it has is that it can help us diagnose bugs that occur apparently randomly (e.g. due to race conditions or timing events). The full execution context is saved, so that once you stumble upon a bug, I can debug it exactly as it occurred in your environment.

6 Installing web2py

6.1 Installing web2py with IIS

It's fairly difficult to give a complete explanation of installing for IIS, mainly because of its peculiar structure with application pools and websites. The method I recommend below is useful for deployment in a testing environment, or in production where you have an already-running application you don't want to stop and reconfigure, and you don't require too much performance. It is also possible to get a full-scale deployment by calling web2py from ISAPI handler in a dedicated AppPool, or with PyISAPIe, but those are difficult to set up and make a lot of assumptions about the deployment environment that I would much rather have the webmaster make.
An outline of all the possibilities is here. The recommended method is the first one – using IIS as proxy and running web2py in its included web server.
http://www.web2pyslices.com/slice/show/1453/install-web2py-on-windows-with-iis
Any of them work just as well.

6.2 Installing web2py with Apache and WSGI

A good tutorial is here:
http://jdoe.asidev.com/2009/02/26/configure-web2py-to-run-behind-apache-with-wsgi-mod_wsgi/
This method is the one I recommend for deployment over a Unix system.

7 Suggested integration points

As I see it right now, there are several places that need serious integration work:
1. User authentication and authorization, which needs to be central for all the applications developed as part of SEETSOC. I think this is really important – having to keep several authentication tokens is not only a burden to the users, it also makes it very difficult to implement a more secure authentication scheme (e.g. with physical tokens).
2. Unfortunately, I have not received any kind of formal specifications regarding the other modules and applications in SEETSOC. I do imagine that some integration has to be performed for computing delivery routes. There is some room left for that in all stages of the controller procedures; for instance, after the merit order is computed, it is not immediately committed to the database –

but rather left for examination (and potential verification by another program).
3. The transaction settlement phase will probably have to be integrated with a central billing module and interface (I don't know what procedure is followed by TSOs, but I imagine they don't just call each other and say "you have to pay this much"). Right now, it saves dispatch orders with what I thought was the required information in a table. There probably isn't any additional step to be taken here in terms of integration, but the format of the table may need to be ammended.
What you need to be mindful of while testing the application is:
1. Wherever validation and verification by an external program was required, it is automatically assumed to be passed. Right now the program will happily accept anything you throw at it, even if it wouldn't pass an external verification – within certain limits (we do run a minimal set of verifications, such as ensuring that quantities have physical meaning).
2. The format of the table and the business and presentation logic can be changed at any time, and with little effort from my side. They're decoupled from the underlying implementation of the algorithms, so we don't risk breaking anything important.

8 Appendix A: Output of History Index view

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>
SEETSOC
</title>

<meta name="keywords" content="" /><meta name="description" content="" /><meta
name="author" content="UPB &lt;alex@zencoding.org&gt;" /><script
src="/seetsoc/static/js/jquery.js" type="text/javascript"></script><link
href="/seetsoc/static/css/anytime.css" rel="stylesheet" type="text/css" /><script
src="/seetsoc/static/js/anytime.js" type="text/javascript"></script><link
href="/seetsoc/static/plugin_layouts/layouts/Incorporated/style.css" rel="stylesheet"
type="text/css" /><script src="/seetsoc/static/plugin_layouts/superfish.js"
type="text/javascript"></script>
<script type="text/javascript"><!--
// These variables are used by the web2py_ajax_init function in web2py_ajax.js (which
is loaded below).
var w2p_ajax_confirm_message = "Are you sure you want to delete this object?";
var w2p_ajax_date_format = "%Y-%m-%d";
var w2p_ajax_datetime_format = "%Y-%m-%d %H:%M:%S";
//--></script>
<script src="/seetsoc/static/js/web2py_ajax.js" type="text/javascript"></script>
<style>
ul.web2py-menu-vertical { width: 150px; position: relative; top:1em; left:1em; zindex:30;
border:0; margin:0; padding: 0; list-style: none; text-decoration: none;}
ul.web2py-menu-vertical li { margin:0; padding:0; border:0; width: 150px; background:
black; text-align:left;}
ul.web2py-menu-vertical li a { margin:0; padding: 1px 3px 1px 3px; border:0; width:
144px; color: white; font-size:12px;}
div.flash { position: fixed; float: right; padding: 10px; top: 10px; right: 30px;
opacity: 0.75; margin: 10px 10px 10px 10px; text-align: center; clear: both; color: #fff;
font-size: 11pt; text-align: center; vertical-align: middle; cursor: pointer; background:
black; border: 2px solid #fff; -moz-border-radius: 5px; -webkit-border-radius: 5px; zindex:
2; }
div.error { background-color: red; color: white; padding: 3px; }
.auth_navbar { text-align:right; z-index:20; position: absolute; top: 2px; right:30px;
border:0; padding:0 }
</style>
</head>
<body>
<!-- start header -->
<span class="auth_navbar">Welcome John [ <a
href="/seetsoc/default/user/logout">logout</a> | <a
href="/seetsoc/default/user/profile">profile</a> | <a
href="/seetsoc/default/user/change_password">password</a>]</span>
<div id="header">
<div id="logo">
<h1>
SEETSOC
</h1>
<p>
Offer Handling Server
</a>
</p>
</div>
<!-- end #logo -->
<div id="menu">



<ul class="web2py-menu"><li><a href="/seetsoc/default/index">Index</a></li><li><a
href="/seetsoc/upload/index">Offer upload</a></li><li class="web2py-menu-active"><a
href="/seetsoc/history/index">Offer history</a></li><li><a
href="/seetsoc/merit/index">Merit order</a></li><li><a
href="/seetsoc/concluding/index">Concluding Transactions</a></li><li><a
href="/seetsoc/settlement/index">Transaction Settlement</a></li></ul>
<script>
jQuery(document).ready(function(){jQuery('ul.web2py-
menu').superfish({delay:400});});
</script>
</div>
<!-- end #menu -->
</div>
<!-- end #header -->
<div id="page">
<div id="content">
<div class="flash"></div>
<h1>Previously uploaded offers</h1>
<table>
<tr align="center">
<td align="center">ID</td>
<td align="center">Ack Time</td>
<td align="center">View</td>
<td align="center">Accepted</td>
</tr>
<tr>
<td align="center">1</td>
<td align="center">2012-04-15 18:16:44</td>
<td align="center"><a
href="seetsoc/uploads/seetsoc_ohs_xmlhistory.filename.adfc2d24cf59bb1c.736f312e786d6c.xml">Click
here to view</td>
<td align="center">
Yes
</td>
</td>
<tr>
<td align="center">2</td>
<td align="center">2012-04-15 18:17:04</td>



<td align="center"><a
href="seetsoc/uploads/seetsoc_ohs_xmlhistory.filename.a60a4dfb1f979043.626f312e786d6c.xml">Click
here to view</td>
<td align="center">
Yes
</td>
</td>
<tr>
```

```
<td align="center">None</td>
<td align="center">2012-04-15 18:22:04</td>
<td align="center"><a
href="seetsoc/uploads/seetsoc_ohs_xmlhistory.filename.97b038aed0a923fb.626f322e786d6c.xml">Click
here to view</td>
<td align="center">
No
</td>
</td>
<tr>
<td align="center">3</td>
<td align="center">2012-04-15 18:22:08</td>
<td align="center"><a
href="seetsoc/uploads/seetsoc_ohs_xmlhistory.filename.91d97fa0be21100a.626f312e786d6c.xml">Click
here to view</td>
<td align="center">
Yes
</td>
</td>
<tr>
<td align="center">4</td>
<td align="center">2012-04-15 18:22:57</td>
<td align="center"><a
href="seetsoc/uploads/seetsoc_ohs_xmlhistory.filename.9d131313ff3029ae.626f312e786d6c.xml">Click
here to view</td>
<td align="center">
Yes
</td>

</td>
<tr>
<td align="center">5</td>
<td align="center">2012-04-15 18:23:13</td>
<td align="center"><a
href="seetsoc/uploads/seetsoc_ohs_xmlhistory.filename.a679046ee643551c.626f312e786d6c.xml">Click
here to view</td>
<td align="center">
Yes
</td>
</td>
<tr>
<td align="center">6</td>
<td align="center">2012-04-15 18:23:29</td>
<td align="center"><a
href="seetsoc/uploads/seetsoc_ohs_xmlhistory.filename.ad665a94bb1abcee.626f312e786d6c.xml">Click
here to view</td>
<td align="center">
Yes
</td>
</td>
<tr>
<td align="center">7</td>
<td align="center">2012-04-15 18:51:09</td>
<td align="center"><a
href="seetsoc/uploads/seetsoc_ohs_xmlhistory.filename.be13b4354fc79e65.626f312e786d6c.xml">Click
here to view</td>
<td align="center">
Yes
</td>
</td>
<tr>
<td align="center">8</td>
<td align="center">2012-04-15 18:52:37</td>
<td align="center"><a
href="seetsoc/uploads/seetsoc_ohs_xmlhistory.filename.a25ad339b39311ea.626f312e786d6c.xml">Click
```

```
here to view</td>

<td align="center">
Yes
</td>
</td>
<tr>
<td align="center">9</td>
<td align="center">2012-04-15 19:14:10</td>
<td align="center"><a
href="seetsoc/uploads/seetsoc_ohs_xmlhistory.filename.b4e130e333a00adc.736f312e786d6c.xml">Click
here to view</td>
<td align="center">
Yes
</td>
</td>
</table>
</div>
<!-- end #content -->
</div>
<!-- end #page -->
<div id="footer">
<p>
Copyright (c) UPB, 2010-2011
</p>
</div>
<!-- end #footer -->
</body>
</html>
```

9 Appendix B: Database Model definition file

```
# -*- coding: utf-8 -*from
repr import rtoi, itor
#########################################################################
## This scaffolding model makes your app work on Google App Engine too
## File is released under public domain and you can use without limitations
#########################################################################

if not request.env.web2py_runtime_gae:
## if NOT running on Google App Engine use SQLite or other DB
db = DAL('sqlite://storage.sqlite')
else:
## connect to Google BigTable (optional 'google:datastore://namespace')
db = DAL('google:datastore')
## store sessions and tickets there
session.connect(request, response, db = db)
## or store session in Memcache, Redis, etc.
## from gluon.contrib.memdb import MEMDB
## from google.appengine.api.memcache import Client
## session.connect(request, response, db = MEMDB(Client()))
from gluon.tools import Auth, Crud, Service, PluginManager, prettydate
auth = Auth(db, hmac_key=Auth.get_or_create_key())
crud, service, plugins = Crud(db), Service(), PluginManager()
## create all tables needed by auth if not custom tables
db.define_table('seetsoc_ohs_parties',
Field('id', 'id'),
Field('Name', type='string', required=True, notnull=True),
Field('timezone', type='integer', required=True, notnull=True))
db.define_table('seetsoc_ohs_xmlhistory',
Field('id', 'id'),
Field('previd', type='integer', required=False, notnull=False),
Field('offer_id', type='integer', required=True, notnull=False),
Field('ack_time', type='datetime', required=True, notnull=True),
Field('eic', type='string', required=True, notnull=True),
Field('accepted', type='boolean', required=True, notnull=True),
Field('filename', 'upload'),
Field('valid', type='boolean', required=True, notnull=True),
Field('entered_transaction', type='boolean', required=True, notnull=True,
default = False))
db.define_table('seetsoc_ohs_offer',
Field('id', 'id'),
Field('eic', type='string', required=True, notnull=True),
Field('delivery_date', type='date', required=True, notnull=True),
Field('creation_date', type='datetime', required=True, notnull=True),
Field('msg_version', type='integer', required=True, notnull=True),
Field('off_type', type='integer', required=True, notnull=True))
db.define_table('seetsoc_ohs_bids',
Field('id', 'id'),
Field('offer_id', type='integer', required=True, notnull=True),
Field('interval', type='integer', required=True, notnull=True),
Field('price', type='decimal(6,2)', required=True, default=None,
notnull=True),
Field('qty', type='decimal(6,2)', required=True, default=None,
notnull=True),
Field('eng_type', type='integer', required=True, default=0,
notnull=True),


Field('bid_type', type='integer', required=True, notnull=True))
db.define_table('seetsoc_ohs_transactions',
# Fields required for transaction settlement programming
Field('id', 'id'),
Field('party_id', type='integer', required=True, notnull=True),
Field('bid_id', type='integer', required=True, notnull=True),
Field('bid_type', type='integer', required=True, notnull=True),
Field('eng_type', type='integer', required=True, notnull=True),
Field('offer_id', type='integer', required=True, notnull=True),
Field('delivery_date', type='date', required=True,
notnull=True),
Field('interval', type='integer', required=True, notnull=True),
Field('qty', type='integer', required=True, notnull=True),
Field('mcp', type='integer', required=True, notnull=True),
Field('orig_price', type='integer', required = True, notnull = True),
# Fields required for transaction settlement adjustment
Field('transactioned', type='boolean', required=True, notnull=True),
Field('final_price', type='integer', required=True, notnull=True),
Field('final_qty', type='integer'))
# Fields required for imbalance prices
db.define_table('setsoc_ohs_settlement_notes',
Field('id', 'id'),
Field('date', type='date', required=True, notnull=True),
Field('interval', type='integer', required=True),
Field('metered_qty', type='integer', required=True),
Field('vimb', type='decimal(6,2)', required=True),
Field('TSO', type='integer', required=True))
db.define_table(
auth.settings.table_user_name,
Field('first_name', length=128, default="),
Field('last_name', length=128, default="),
Field('email', length=128, default=", unique=True),
Field('TSO', type='integer'),
Field('password', 'password', length=512, readable=False, label='Password'),
Field('registration_key', length=512, writable=False, readable=False, default="),
Field('reset_password_key', length=512, writable=False, readable=False, default="),
Field('registration_id', length=512, writable=False, readable=False, default="),
format='%(first_name)s %(last_name)s')
member = db[auth.settings.table_user_name] # get the custom_auth_table
member.first_name.requires = \
IS_NOT_EMPTY(error_message=auth.messages.is_empty)
member.last_name.requires = \
IS_NOT_EMPTY(error_message=auth.messages.is_empty)
member.password.requires = [IS_STRONG(min=5, special=0, upper=0), CRYPT()]
member.email.requires = [
IS_EMAIL(error_message=auth.messages.invalid_email),
IS_NOT_IN_DB(db, 'auth_user.email')]
auth.define_tables()
```