



Apache OFBiz development Introduction.

1 Table of Contents

1 Introduction to this development introduction.....	3
2 Overview OFBiz development.....	3
3 Local Installation: development environment.....	4
4 Staging/Production Environment.....	5
5 Framework introduction.....	5
5.1 Entity engine.....	6
5.1.1 Entity Engine guide.....	6
5.1.2 Entity Engine Configuration Guide.....	6
5.2 Service Engine.....	6
5.3 OFBiz minilanguage.....	6
5.4 Security.....	6
5.5 OFBiz widgets.....	6
5.6 Help screens & internal documentation.....	7
5.7 Birt report generator.....	7
5.8 Portal functions.....	7
5.9 Business Intelligence.....	7
5.10 The java Api.....	7
6 Bulk upload of data.....	7
7 Entity definition.....	7
8 Application introduction.....	7
9 Create the custom component in hot-deploy.....	7
10 Create company demo data.....	7
11 Change OFBiz itself.....	8
12 Coding rules.....	8
13 Testing.....	8
14 Continuous testing environment.....	8
14.1 Installation.....	8
14.2 Create and configure Hudson.....	9
15 Documentation.....	9
16 Version control with SVN.....	9
17 Project mailing list.....	10
18 Integration with other systems.....	10
18.1 Available integrations incorporated in OFBiz	10
18.1.1 CAS	10
18.1.2 LDAP	10



18.2 Database level	10
18.3 Web service level	10
18.4 User interface level	11
18.5 Replication and copying	11
19 Do the practice exercises.....	11
20 Scrum methodology.....	11



1 Introduction to this development introduction

To get started with any complex system is not easy. This document is an introduction to get started quickly in a professional OFBiz development environment, being able to set up the system locally and making your first modifications and still be able to upgrade easily. Also some guidelines how to install your changes in a test/staging environment and how to move them to a demo/production environment.

Please also remember that development with OFBiz is a lot different from other environments for the following reasons:

1. Framework.
Extensive framework making the system independent from database and operating system and provides rapid development using a custom mini language with screens and forms. Provides a SOA environment both in foreground and background. No detailed technology knowledge required such as SQL, HTML, Javascript etc for general business functions.
2. Every information item or business function is only implemented once, which means no duplication in data and/or programs. If an error is discovered, it only need to be fixed once.
3. Business entity relationship diagram (re)usage:
This data model is based on the Data model and resource books from Len Silverston and provides more than 800 entities and 200 views which are fully normalized.
4. Business application service level ERP and CRM functions (re) usage:
Making use of the framework and entities, more than 2000 services are present in the system to update and when the data s complicated for the retrieval of data which can be (re) used.
5. Business application user interface ERP and CRM functions.
On top of the framework, data model and services an extensive application is developed which provides an user interface which can be re-used in custom developments.

So development with OFBiz requires knowledge of the framework, entity data model, services and application interface to fully exploit and reuse its capabilities. This means that programming new functions hardly happen, most of the work is in finding a function which comes close to what you want and then modifying it to your requirements.

2 Overview OFBiz development.

As mentioned in the introduction, most of the work on OFBiz is to investigate if the to be developed function is already present and if the available functions are sufficient. When this is determined, most, if not all the changes need to be created in a newly component in the hot-deploy directory. New functions will also be stored there.

Sometimes OFBiz itself needs to be changed too. For these changes, patches will be created for a certain version of OFBiz and they will also be kept in the newly create component. More details about this later.

In this way OFBiz itself will not be changed to enable the upgrade to future versions what is important



in the quickly changing world both in business and technology.

For every addition or modification it needs to be decided if the new development is beneficial for other OFBiz users. If so, a contribution can be made according the guidelines at <https://cwiki.apache.org/confluence/display/OFBADMIN/OFBiz+Contributors+Best+Practices> .

Why? First of all, to contribute back as a 'thank you' for the existing system to make it better and secondly because also your changes or additions will be maintained which will save you work.

It is advised to always use the latest trunk version for development and update the system regularly, at least every week in development and at least every 3-6 months in production.

3 Local Installation: development environment.

To do proper development it is required that you install the system locally. You need at least 1.5Gb of memory to run Eclipse (the preferred IDE) and OFBiz simultaneously.

You can either develop under Linux (Ubuntu or other flavors), Apple Mac or Windows and other operating systems which support the Java runtime.

If you are using Ubuntu, like we do, almost all applications can be installed using the Synaptic package manager in administration. Make sure you have enabled the third party repositories under preferences/repositories otherwise the Java JDK will not show. The links below are only provided for non Linux users.

The applications you need:

1. Eclipse, at <http://www.eclipse.org>, select the java version. (at the moment the newest Eclipse contains the XML editor and is therefore preferred above the version in the Synaptic package manager)
(To debug OFBiz use remote debugging. Uncomment the debugging line in startofbiz.sh/bat and comment the existing startup line. This will open port 5005 for remote debugging. Start OFBiz and then go to eclipse -> debug configurations -> new remote java application Enter a name, select the correct project and change the port to 5005. Click apply and then debug.)
2. Apache Subversion client either on the command line or inside Eclipse.
(<http://subversion.apache.org>)
3. A program called "patch" in order to apply patches against OFBiz. (for windows users: <http://unxutils.sourceforge.net/>)
4. A ssh client to access the test/demo environments. (Putty in Windows)
5. The Sun java 6 SDK (<http://java.sun.com/javase/downloads/widget/jdk6.jsp>) , OpenJdk does not yet work.
6. OFBiz , checkout the lastest trunk version for development. More info at: <https://cwiki.apache.org/OFBADMIN/ofbiz-source-repository-and-access.html>



Checkout svn command for read access:

svn co <http://svn.apache.org/repos/asf/ofbiz/trunk> ofbiz

On-line browsing of the repository: <http://svn.apache.org/viewvc/ofbiz/trunk/>

After you have installed all these applications, go to the OFBiz home directory and initialize the system as is described in the README file in the OFBiz home directory. Then you should be able to see the OFBiz system in your browser!

4 Staging/Production Environment.

For staging and production 2 different separated OFBiz installations are required

Preferably a Linux server is used to run these environments. For the second environment the ports need to be adjusted if you are running in the same server. A good document here is the product setup guide at:

<https://cwiki.apache.org/confluence/display/OFBTECH/Apache+OFBiz+Technical+Production+Setup+Guide>

For running 2 OFBiz systems you need at least 4Gb of memory.

An other alternative is using the recently introduced 'multi-tenant' functionality which is described at:

<https://cwiki.apache.org/confluence/display/OFBIZ/Multitenancy+support>

Using this functionality you can run two different OFBiz environments on the same system.

After the OFBiz system checkout from SVN you can install your local hot-deploy component (if any), apply the OFBiz patches, run the seed update ant task and (re)start the system.

If you would later on update the system use the 'SVN up' command and rerun the patch and seed loading programs.

Also setup a script to run the system tests every night on your test environment.

To shutdown the OFBiz system execute the 'stopofbiz.sh' command repeatedly until a connection error appears which shows the server has stopped.

5 Framework introduction.

For an introduction please visit the following document and the option 'webtools' inside the ofbiz application.

<http://cwiki.apache.org/OFBTECH/framework-introduction-videos-and-diagrams.html>



5.1 Entity engine.

5.1.1 Entity Engine guide.

[Entity engine Guide](#)

5.1.2 Entity Engine Configuration Guide.

[entity engine configuration guide](#)

5.2 Service Engine.

Service engine description:

<http://cwiki.apache.org/confluence/display/OFBTECH/Service+Engine+Guide>

Any service can also be called as a web service. And example: (using a soap client)

<http://demo-trunk.ofbiz.apache.org/webtools/control/SOAPService>
with the following message:

```
<ping>
<map-Map>
<map-Entry>
<map-Key><std-String value="message"/></map-Key>
<map-Value><std-String value="pong"/></map-Value>
</map-Entry>
</map-Map>
</ping>
```

Outside going web services can be used at the same way as internal web services are used.

5.3 OFBiz minilanguage

<https://cwiki.apache.org/confluence/display/OFBIZ/Mini-Language+Guide>

5.4 Security

<https://cwiki.apache.org/confluence/display/OFBTECH/OFBiz+security>

5.5 OFBiz widgets.

<https://cwiki.apache.org/confluence/display/OFBIZ/Understanding+the+OFBiz+Widget+Toolkit>



5.6 Help screens & internal documentation.

5.7 Birt report generator.

5.8 Portal functions.

5.9 Business Intelligence.

5.10 The java Api.

<http://ci.apache.org/projects/ofbiz/site/javadocs/>

6 Bulk upload of data.

A spreadsheet example is at: \applications\product\src\org\ofbiz\product\spreadsheetimport

7 Entity definition.

An overview of the entity relationship diagram can be found here:

<http://cwiki.apache.org/confluence/x/8ILK>

8 Application introduction.

A full ERP/CRM system is included and an overview is given in the document: “OFBiz introduction” which can be found on the same site you found this document.

9 Create the custom component in hot-deploy

Updates start with creating a new component in the OFBiz hot-deploy directory with the command:

```
./ant create-component
```

This will create a component in the hot-deploy directory.

10 Create company demo data.

On order to create specific company demo data a component is created called “setup”. This component will however only show if only seed data is loaded. Please look at the setup helpscreens in the help system for more information.



11 Change OFBiz itself.

When the custom component is created a number of new targets will be in the new components home directory you can list with following command: `../../ant -p`

this will list the following:

`apply-ofbiz-patches` Apply the patch to framework, application, specialpurpose components.

`create-ofbiz-patches` Creates patch for framework, application, specialpurpose components

`reapply-ofbiz-patches` First removes any previously applied patch and then applies the new patch

`revert-ofbiz-patches` Remove any local change in the files or any previously applied local patch.

This will enable you to simply change OFBiz and with the above command automatically save the patch in the custom component which can be committed to SVN.

12 Coding rules.

If you would like to follow the coding conventions which are currently used within the system checkout the coding rules:

<https://cwiki.apache.org/confluence/display/OFBADMIN/Coding+Conventions>

13 Testing.

In OFBiz we use 2 different ways of testing: at a service level (JUnit) and at the user interface level (Selenium). For the examples see the system itself and check the help screens.

JUnit testing is explained in the helpscreens or in the ofbiz document with the same text:

http://demo-trunk.ofbiz.apache.org/cmssite/cms/APACHE_OFBIZ_HTML#N2327C

14 Continuous testing environment.

It is advised to install an nightly job which will update the current installed version, compile it and run all tests overnight and log the result. The first job every morning is to fix the reported errors if any. Apache Hudson is the preferred tool here.

14.1 Installation.

1. Download hudson from <http://www.hudson-ci.org/latest/hudson.war>
2. Run The Hudson by using command `#java -jar hudson.war --httpPort=8086`



14.2 Create and configure Hudson

1. create a new project and choose a name for the project. Select "Build a free-style project" for the project
2. Configure
 1. Use custom workspace: Directory = path of project
 2. Subversion: Repository URL = Repository of project, Local module directory (optional) = project directory
 3. Use update: true/false
 4. Poll SCM: Schedule setting
 5. Build: Execute shell = command line what want to do.
 6. Post-build Actions: Publish JUnit test result report >> Test report XMLs = runtime/logs/test-results/*.xml

15 Documentation.

Ofbiz documentation is contained in the system itself:

1. user documentation: help screens of which a document can be generated.
2. Service documentation: in the service definitions, shown in webtools.
3. Entity documentation: in the entity definitions, shown in webtools.
4. General documentation in the OFBiz wiki at ofbiz.apache.org.

16 Version control with SVN.

Do not keep OFBiz itself in your local SVN, only your hot-deploy component. Keep track of the current used version in a README file in the component root directory. Store the hot-deploy component in your local SVN.

Use the following configuration file:

<http://SVN.ofbiz.org/SVN/ofbiz/trunk/website/SVN/config> (change for new wiki)

useful SVN commands:

checkout (ofbiz):

```
svn co http://svn.apache.org/repos/asf/ofbiz/trunk  
for a specific revision use the "-r xxxx" option.
```

Update to the latest version:

```
svn up
```



for a specific revision use the “-r xxxx” option.

Before you commit to SVN always check first with the following commands if that is really what you want to change, Some times you can mis-type and did not notice it.

See what has been changed:

```
svn st
svn diff
```

Or put the changes in a file: “SVN diff >changes.diff” and use an editor to browse them.

If you are sure that this is what you want to change, enter the command:

```
svn commit -m “log message”
```

Make sure you have a meaningful log message referring back to the original customer request.

You can optionally add a directory or filename to the last command.

The SVN book free to download: <http://svnbook.red-bean.com/>

17 Project mailing list.

18 Integration with other systems.

18.1 Available integrations incorporated in OFBiz

18.1.1 CAS

18.1.2 LDAP

18.2 Database level

Ofbiz can access different databases at the same time and can read databases from other systems. In this way data from other systems can be integrated and the service and presentation layer of OFBiz can be used on foreign data.

18.3 Web service level

OFBiz is using Apache Axis 2 for the delivering of web-services. (<http://ws.apache.org/axis2/>) and in principle has all the functionality Axis 2 has. It supports the international web standard defined at: <http://www.w3c.org>. To access web-services from the outside of OFBiz the following is possible: (using the demo OFBiz system as a target)



To get started with OFBiz services you need the following OFBiz specific information.

Call the OFBiz url with the following mandatory parameters: serviceName

example service: “ping”:

<http://demo-trunk.ofbiz.apache.org/webtools/control/httpService?serviceName=ping>

You can try this in a ordinary browser and you will be send the result in a separate document.

To obtain the WSDL use the following example:

<http://demo-trunk.ofbiz.apache.org/webtools/control/SOAPService/ping?wsdl>

(use the source page view) to show the wsdl xml document.

A list of possible exported services can be obtained by:

<http://demo-trunk.ofbiz.apache.org/webtools/control/SOAPService/?wsdl>

18.4 User interface level

Systems can be integrated at the user interface level using html Iframes or using the OFBiz screens feature.

18.5 Replication and copying

As is possible in any system to system interface you could generate a file from OFBiz and make it available to the other system and vice versa at regular intervals. In OFBiz you could create a service which starts in the background at regular intervals and read and/or create the interface file.

19 Do the practice exercises.

To actually practice the framework functionality checkout the exercises at:

<http://cwiki.apache.org/confluence/x/cQFk>

This exercise give you an introduction into the framework by creating your first new component in the OFBiz framework.

20 Scrum methodology

This is an optional development requirement using Scrum, we do however advise you to have a look. There are a number of good short documents describing the methodology:

Scrum guide: <http://www.scrum.org/scrumguides/> (various languages)

Scrum in 30 seconds: http://www.scrumalliance.org/pages/what_is_scrum

Scrum in 5 minutes: http://www.softhouse.se/Uploades/Scrum_eng_webb.pdf

We are currently developing and OFBiz component which supports the scrum methodology so you can use Scrum within an ERP system integrated with your employees, customers and accounting.



Some of our scrum experiences.

1. If you are developing with OFBiz especially in the outsource situation a web based system is essential so your customer can see what is happening, where the hours are used etc. OFBiz-Scrum is a good example of that.
2. Manual testing should be done as early as possible. Make sure there is always a task in a backlog item where another team member will test the item according the backlog definition as soon the implementer has finished the item. Error tasks added by other team members will always have priority above the implementation of new backlog items. If implemented the tester will test again until all tests succeed.
3. Make sure at the start of the sprint that all items in the sprint backlog are understood by all members of the team.
4. Make sure everybody understands the definition of “done”. In the case of ofbiz we have the following list:
 1. the actual implementation
 2. The automated test including security/roles tests
 3. Help screens or explanation in the internal OFBiz document.
 4. Describe how to test the function, next to the automated tests. (where to store?)
 5. Demonstration data.
 6. Test task by a different team member. Testing includes the review of the automated tests and usefulness of the screens, fields and functions (if any)
 7. All reported errors should have automated tests assigned to them to make sure it does not happen again. (when possible)
5. Anybody can add 'error' tasks to a backlog item when anybody in the team find an error and to keep track of them.
6. When checking work of other members of the team focus on the following subjects:
 1. Is the code following the OFBiz coding rules mentioned earlier in this document.
 2. Are the automated tests sufficient, are the different roles tested?
 3. Is the change implemented according the backlog item?
 4. If errors are found report these back a a new task on the backlogitem.

TODO: introduce selenium tests for the user interface.