OFBiz Contributors Best Practices

Why You Should Contribute to OFBiz

By contributing your improvements back to OFBiz, you can get our entire community of developers and users to help you debug, improve, or extend the features that you need for your business. Furthermore, if your contributions improve OFBiz, then it would help to attract more users and more developers for OFBiz down the road, and eventually those users and developers would make contributions that would benefit you. Finally, the process of contributing back to the project is a great way for new users and developers to work with the existing community and learn more about OFBiz so they could tap into its power and flexibility.

How to Contribute to OFBiz

OFBiz is a community-developed open source project. What that means is that we're looking to you, the user, to help make our application better. Anybody can contribute to OFBiz, you do not have to be a committer, on an "approved" list, or be a friend or relative. All contributions are considered based on their merit for the project. You also do not need commit privileges to make a contribution. Just create a patch file and post it on our JIRA issue tracker or/and ask for being a wiki contributor.

For bug reporters

Bug reports are important and welcome, and people take them seriously even when things aren't clear. Part of the problem is that OFBiz is a big system and without adequate details it's easy to do something completely different than what you did. To help make them clear you might try including the following:

1. What you did (including detailed steps to reproduce, with URLs, field names, exact quotes of labels, etc)
2. What you expected to happen
3. What actually happened (again including exact quotes of error messages, etc)

There are, at least, two kind of contributors: bug reporters, and plain contributors (the ones who follow these Contributors Best Practices). If you don't have enough time or competence to solve a bug by yourself, you may simply report it on the user mailing list. But then please respect the conventions above.

When to create a Jira issue

1. Before creating any Jira issue, please check, using some related key words, if a similar issue does not exist already. For that you can first use the quick search at top right of Jira pages and after refine your search using relevant information. For instance by default all projects are scanned, you may then search only in OFBiz, etc.
2. When you find a duplicate of a JIRA issue, please close the newer one with the resolution 'Duplicate', and state what the original issue is. This does not apply for placeholder issues or if the newer Jira does contain additional aspects or more comprehensive solutions, documentation or patches. If in doubt, please discuss on the development mailing list before choosing one issue over the other.
3. If you already have a patch for an improvement/fix then create a Jira issue (if there is not one already) and attach your patch to it.
4. If you don't have a patch, and you have discovered a **bug**, create a Jira issue (if there is not one already) providing as much details as possible (including the rev. number and the environment you are using, and the step to recreate the bug). if you have no ideas how to describe the bug use this template:
    a. What you did (including detailed steps to reproduce);
    b. What you expected to happen;
    c. What actually happened (including exact quotes of error messages, etc);
    d. If possible provide an URL from one of our demo sites.
5. If you don't have a patch, and you have want to suggest an enhancement or new feature, then discuss this in the dev mailing list instead of creating a Jira issue; at the end of the discussion, the community will consider if a summary of the thread should be saved in a new Jira issue, to facilitate future development.
6. If you don't have a patch, but you are planning to work on it, and you want to share your design details with the community, you should discuss this in the mailing list instead of creating a Jira issue; if, on the other hand, you don't have time to do this, you have already decided that you want to implement your patch following your design notes, and you just want to let the community know about the upcoming patch, you can create a Jira issue (to which you will attach your patch when it is ready);

    Summarizing:

- Bugs: always create a new Jira issue everytime you find a new bug
- New features/enhancements: create new Jira issue only if you have a patch (or if you plan to submit it very soon).

How to create a Jira issue

1. Create an account here, if you do not have one
2. Login
    a. (optional if you are sure it's new) Search if an issue for what you are after already exists by using the "Find issues"
    b. (optional if you are sure it's new) If an issue on the subject already exists you can add a comment on it
3. If a issue does not exist, create a new one selecting the "create new issue" command. For details on the issue creation see here
4. Select the OFBiz project and the issue type.
5. Fill in all fields, give as many detail you think necessary
    - Generally it is very important to select in the "Affect version" field the OFBiz version you are running.
    - Use the Environment field to specify at least your operating system and the database, OFBiz is using since these information could be very useful to help people to work on the issue. If you are using the trunk branch for your development then the SVN revision should also be specified in the Environment field.
    - Select the concerned component(s). If all components are affected select ALL_COMPONENTS (uncommon case)
    - Leave the "Fix Version/s" field empty, unless you are adding something new (Improvement or New Feature) then "Upcoming Release" is fine. This field will be used by committers to specify the future release that will contain the fix.
6. If you need to attach files such as patches you must do it as a second step after the issue creation. It is also possible to easily attach screenshots to the issue see here
    - When attaching files or screenshots you can add a comment where you explain how the attached file is supposed to be used. Please reference the file name in the comment because more files could be attached to the issue at a later time and they will be all listed together far away from their comments. If, for any reason, you don't want your patch or attachment to be granted to the ASF or committed, please note it in one related comment (possible cases: not ready yet, examples, etc.)
    - Also please use preferably .patch as extension for patches. When updating an attached file keep the same name : Jira is able to deal with that and will simply gray old files, you don't need to delete them (sometimes its useful to compare older patches versions)
    - If you provide a patch, be sure to use the button "Provide Patch" (the status will then change to "Patch Available"). This allows us (committers, and other contributors) to know that this issue is ready for review.
7. Jira offers a voting mechanism that is used to give more relevance to the issues (see here to learn more).

How to work with Confluence (by order of preference)

1. If you are a registered wiki contributor (recommended) you can edit any page in the wiki, just be sure to add a sentence about what you did when committing.
2. If you are not a registered wiki contributor and don't want to be one, you can still add your Confluence formatted contents as comments in existing pages. So if you need a new page you need to ask for its creation before adding comments...
3. If you don't want to provide Confluence formatted user stories then open a Jira and add your unformatted user stories in a text file

Following changes

Please **read** this section in OFBiz Committers Roles and Responsibilities page

Editing comments in Jira

This feature should be used very parsimoniously because it's not easy to read edited comments from a mailing list and most people read comments from the dev ML (Jira issues are redirected to dev ML).
So, as far as it's possible, you should better add a new comment than editing one. If you **really need** to edit a comment, you **MUST** put a **BIG prefix** before your comment so it is possible to distinguish it from the original text. That should include more than just a pair of "*" to bold part or all of the response, and should also include your initials so that it is clear which things you added.

If you become a frequent contributor and are willing to help with the long term development of the project, you could become a project committer.

The following guidelines are meant to help contributors work with the committers and the community as a whole:

Guidelines

1. Follow coding conventions. **Seriously, read that document**.
2. Install the OFBiz Subversion client configuration file
3. Follow the 2 main rules for committers:
   a. **Rule #1 for a committer is the same as for a doctor: first do no harm**. Nothing should be committed that breaks existing functionality without replacing it either before or in the same commit. Whatever you are working with someone developed it and chances are someone is using it, and possibly MANY people.
   b. **Rule #2 for a committer is the same as for a scientist: read before you write**. When you're getting started a good time ratio for read to write is around 20 to 1. Once you're a total OFBiz pro who knows as much as any living person about the project, you can probably reduce that to about 3 to 1.
4. Discuss your features with the community. What are you trying to implement, and how are you planning to do it? This is especially important if you are new to the project.
5. Write clear, well-commented and understandable code. Don't take shortcuts, especially around variable names and error or warning messages. Use understandable data structures. Remember, somebody else will be working with your code down the road.
6. When you prepare a patch, do your best to avoid to mix formatting changes with relevant changes (if possible, provide a separate patch containing only formatting changes): in this way the reviewer's work will be easier
7. When you prepare a patch, do not insert in the code comments with author information since your name will be recorded in the commit log (that is the place were we store this kind of information)
8. Internationalize your code with UI labels.
9. Start out with small contributions which are easier to review. In the process, get familiar with the project's coding style and "thought process."
10. Keep patches and contributions easy to review and commit. Even if a lot of code is touched, try to keep things isolated and the intent of the patch (es) clear. Remember that most committers can find 20 minutes here and there, but it is very hard to fit in the 2-4 hour time block required to review and commit a larger patch (especially if it touches ANY lower level or more sensitive or complex artifacts, and this requires more thorough review).
    Also try to give committers enough information to quickly test your patches if it's a bit complex
    a. For instance steps to test
    b. If possible an URL, it's always helpful
11. Put your contributions on JIRA instead of emailing it directly to the committers, so everybody can review and comment on it. Though it is not necessary this makes contributions more traceable for the licensing through the Apache Software Foundation.
12. Get other members of the community to try your patch. Write the dev list and tell them what you've done and ask them to try it out and vote for it. This will help the committers when they are reviewing your patches as there will be more confidence that the patch does what it is intended to do, and doesn't break anything else.

If you are planning a larger contribution, please follow the following tips to facilitate both licensing and collaboration. These tips will make it easier for committers to review and incorporate your work, and will overall speed up your development process as you'll be asking the OFBiz team to do a number of small, simple tasks rather than a couple of large tasks that a committer will have to find significant time to review and commit.

Tips

1. Do not implement large blocks of artifacts (code, etc) on your own and then contribute them to OFBiz.
2. If you have a large block of code to contribute we can work with you on that, but it requires a different review and legal vetting process than normal contributions, as described on http://incubator.apache.org/ip-clearance/index.html.
3. When a method is deprecated, it should be explained which method should be used, and what has changed.
4. Instead develop and contribute as you go. This means develop as you would normally, but interact with the OFBiz community through mailing lists and contribute patches regularly. # If you are do not have a committer on your team this can slow down development, so do what you can to "sell" one of the committers on your project and get an ally on the committing team to regularly review and commit your patches. Note that if you let us know in advance that you are planning a larger effort of this nature, we can perhaps find a volunteer beforehand to work with you on this.
5. Just please remember that there is no paid staff on OFBiz, all are volunteers. You may see your patch sit in Jira for a long time while committers work on other things. This usually happens because a committer is working on a priority for the project that has been a problem for a while, or on a paid contract in order to survive and to be able to continue helping with OFBiz.
6. It might be tempting to run your effort without getting an OFBiz committer involved, but keep in mind that committers can help you with technical, business, and legal concerns either on their own or through collaboration with others in the project on in the ASF more generally.

Naming entities and their fields

Here are some conventions, or say patterns, which are common for defining an Entity and its Fields.

- Entity names must be in UpperCamelCase.
- Entity names must be short enough so that the automatic table name (with an underscore added before each capital letter) is 30 characters or less.
- Field names must be short enough so that the automatic column name (with an underscore added before each capital letter) is 30 characters or less.
- fk-name must be 18 characters or less.
- If entity name is abbreviation like Unit Of Measure (UOM) then treat it as one word, like: Uom.
- The field name must be in lowerCamelCase and the name should be self descriptive enough to show the purpose of the field.
- If relation tag specify the relationship between two entities then the fk-name should contains the words from both entities separated by ("_") underscore.
- If a entity relation with another entity defines more than one time then it should be differentiated by title attribute while defining a relation like "From" or "To".
- In the same if both fields in the <key-map> tag are the same, then there is no need to specify the rel-field-name.
- In case of view entities the name must consist of names of all its member entities.
- The <view-link> should be defined for proper view in the webtools.

Apart from the above you can also refer to entitymodel.xsd for understanding the tags.

Deprecating entities

Whenever we deprecate an entity in OFBiz there are certain things that MUST be done or all committers should reject the patch:

1. rename the entity to deprecate by adding an "Old" prefix to it, then specify a table-name attribute on the entity so it still points to the same table in the database
2. create a new entity the replaces the old one, and comment on that fact
3. implement a service to move data from the old/deprecated entity to the new one

You'll see this pattern used in a few places. This is kind of the way that users in general have some sort of hope of being able to update from one revision of OFBiz to another.

Deprecating entities : more information here

How to send in your contributions (or how to create patches, even with Git)

The first step is to create an account for yourself on the JIRA issue tracker and then create a new issue. Describe the contribution that you are making: are you fixing a bug, improving an existing feature, or creating a new feature? Please write as detailed a description as you can and attach screenshots if possible of what you've done. OFBIZ is a huge project, and what may be obvious to you might not be to someone else, even a committer who is familiar with the project.

Then, send in a patch file. This is a file which will describe all the differences between your modified code and the code in the OFBiz Subversion repository. Please use the following conventions to name your patch.
The patch file name should be **OFBIZ-number_featureDescription.patch** where OFBIZ-number is the name of the Jira issue and featureDescription is full Jira issue title if it's small, or part of title if it's big.

You can create a patch file by using

- Command line (see the howto below)
- Eclipse internal command (don't use finish but rather **select project** to avoid the 2 1st lines in the patch)
- A tool like Tortoise
- **If you use Git to create you patch please use "git diff --no-prefix" option to create it.**

No patches when moving files

Patches must never be used to move files. Else if applied we not only lose history when doing so, but also annotations.
The diff and patch commands (even "svn di" and "svn patch") are unable to keep the information about moved files, only deleted and created.

So if you are moving files, simply create patches if some files refer to the paths of the relocated files (and hence need to be changed) and tell the committers which files should be moved (from -> to)
Trailing white spaces

Please adjust your IDE or similar to not remove trailing white spaces when you create a patch, especially a big one, you may do that temporarily. Consider that else reviewers have to be confronted with a lot of false changes, thanks!

We prefer that you build your patch from the root as it's easier for committers to merge your change. So please make your patch files from the OFBIZ_HOME directory (usually ofbiz/) and with relative file paths.
"Relative file paths" means the in your patch file names should look like these :
applications/party/widget/partymgr/PartyScreens.xml
framework/webtools/config/WebtoolsErrorUiLabels.properties
and should not have file names like: C:\myfiles\ofbiz\applications\party\widget\partymgr\PartyScreens.xml

Examples using command line:

```
svn di applications/product > OFBIZ-number_featureDescription.patch
```

If you have added new files, then use the "add" command first, then make the diff

```
svn add applications/product/<my-file>
svn di applications/product > OFBIZ-number_featureDescription.patch
```

You can also specify the exact files that you'd wish to include in your patch file, in case there are files that you have modified but do not wish to submit. For example, you can use

```
svn status applications/product
```

to see which files have been modified (they start with an "M") or added (which start with a "?").
Then do:

```
svn di applications/product/entity/ applications/product/script/org/ofbiz/shipment/shipment/ShipmentServices.
xml > OFBIZ-number_featureDescription.patch
```

if you only want to make a patch file of the entity/ sub-directory and the ShipmentServices.xml file.

Make sure that the from/current revision of your local sandbox (checkout) is the current revision, or a very recent one, from the OFBiz SVN repository. The local revision can be checked by doing:

```
svn info
```

To make sure you have the most recent revision always do an SVN update before doing the patch with svn diff, something like this:

```
svn up
```

This must always be done before submitting a patch otherwise the patch just won't work. If your local sandbox is checked out from a separate SVN repository following the vendor branch pattern instead of directly from the OFBiz SVN, then you should do a vendor branch update, merge, and then local update in your sandbox before doing the svn diff to create the patch.

Next upload your patch file to your JIRA issue. Please use .patch as extension, some tools use extensions and this facilitates committers works.
Finally, if there are several patch files already on an issue, please write a comment about which file should be used. A best practice is also to keep the same filename if a patch file is updated. Jira will take care of that and will keep (better for history) but gray deprecated files with the same filename. It's easier for committer to see at 1st glance which file to use. You can read in 1st comment below what may happen when not using the same filename.

When a Jira issue is totally resolved, we prefer to close the issue than to put it as resolved. There are some cases where there is a tentative resolution and you don't want to close the case because you want the reporter or someone to review and test the fix to make sure it was what was intended. If it is a simpler issue and especially if you are the reporter, then it is best to just close it right away rather than coming back to it later to close.

Why is it Taking So Long to Get My Patch into OFBiz?

The first thing to remember is that in order for something to get done an individual or group must have sufficient ability and time to do it. There are no paid OFBiz committers, everyone works on a volunteer basis. This is a natural side effect of OFBiz being a community-driven project rather than "commercial open source".

When someone submits a patch they are asking for someone in the group of committers to do something for free. Sometimes because of the volume of patches it is overlooked and then with a constant stream of new issues, complaints, questions, etc it may be a long time (if ever) before someone gets back to it. Most unfortunately there aren't any committers that can work on contributing to OFBiz full-time because so far there are no committers that don't need to also earn a living. Most committers work with OFBiz in their day job, and because of the size and complexity of the project so far that seems to be the only way someone even can consistently contribute to OFBiz. That doesn't mean they are paid to work on your issue though, unless you and they get lucky and it happens to be related to a paid client objective.

If you REALLY want your patch in, you can get it in. If you want it in, your job is to help the committers get it in. You can recruit others on the mailing lists to review and test your patch. This is really important, because OFBiz is fairly complex and many patches break rule #1, which is in short, "first do no harm". In other words, do break stuff that already exists that other people implemented, and that other people are *using*.

There is also another option... as mentioned above OFBiz is unfunded and every committer has an employer of some sort, often a handful of clients. This would explain why things are getting committed all the time, even though sometimes the volume of patches getting reviewed and committed is pretty low. If committers are lucky then they get to work on stuff that goes back into OFBiz, and that is ONLY reason that most of the functionality in OFBiz exists at all, especially in the applications (most of the framework, on the other hand, was not ever sponsored). Some people see this as unfair. Others see it as a great stroke of luck that they can take advantage of.

If this isn't working for you, then consider getting more involved with OFBiz or somehow making it possible for others to get more involved with OFBiz. There are many ways you can help, even without becoming a committer yourself. There are the exact same things you can do in order to become a committer if you do enough of them, but of course you can do all of these without any longer term commitment or hoops to jump through. You don't even have to get a committer or PMC member to do anything in order to do these things!

1. Subscribe to the dev mailing list, try to read the majority of the messages, and participate in discussions there
2. Review and comment on issues in the Jira issue tracker
3. Apply patches from Jira locally and test them and comment on the results
4. Create patches to fix issues reported in Jira
5. Get to know OFBiz and submit patches to fix problems or annoyances you find
6. Follow all of the rest of the advice in this document

How Do I Become a Committer Myself?

If you're interested in becoming a committer, that's great! We would love to have you and the whole community will really appreciate your help.

Being a committer can be a great help to your employer and/or clients. It can also be a great asset in your personal and career growth. There is a guarantee that in the course of your activities you will learn and grow. You'll learn about building enterprise applications, both the technical and business sides of them. You'll also learn a lot about working with other people, especially working with other people remotely.

For more information and to get started see the Committers Roles and Responsibilities page.