|  |  |
| --- | --- |
|  | Code Committer Guide  Version 2.0.Beta1 - April 2011 |

Abstract

This document describes how to contribute code as a committer to the Microsoft Biology Initiative which encompasses the Microsoft Biology Foundation (MBF) and Microsoft Biology Tools (MBT).

For updates to this document and the rest of the MBF documentation, see   
<http://mbf.codeplex.com/documentation>

For updates to MBF, see <http://research.microsoft.com/bio>

For community forum questions, see <http://www.getsatisfaction.com/mbi>

Contents

[Introduction to participating as a Committer 2](#_Toc288728911)

[About committer code contributions 2](#_Toc288728912)

[Become a Committer and Partner 3](#_Toc288728913)

[Committer code review responsibilities 3](#_Toc288728914)

[Install TortoiseSVN 4](#_Toc288728915)

[The MBI committer contribution process 4](#_Toc288728916)

[Committer getting started 5](#_Toc288728917)

[Follow on-boarding instructions 5](#_Toc288728918)

[Sign up for MBF email and discussion groups 5](#_Toc288728919)

[Use available resources 6](#_Toc288728920)

[Dependent software products 6](#_Toc288728921)

[Review the C# coding and documentation guidelines 7](#_Toc288728922)

[Review existing code 7](#_Toc288728923)

[Create a new work item 8](#_Toc288728924)

[Write new code 8](#_Toc288728925)

[Write documentation for new code 8](#_Toc288728926)

[Write unit tests for the new code 9](#_Toc288728927)

[Run tests 9](#_Toc288728928)

[Run FxCop 9](#_Toc288728929)

[Apply and review a Contributor patch 10](#_Toc288728930)

[Prepare a Shelveset for code review 10](#_Toc288728931)

[Request code review from committers 11](#_Toc288728932)

[Respond to code review recommendations 11](#_Toc288728933)

[Submit updated shelveset for review 11](#_Toc288728934)

[Submit the changes to the code base 11](#_Toc288728935)

Disclaimer: This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2010 Microsoft Corporation. All rights reserved.

Microsoft and Windows are trademarks of the Microsoft group of companies.  All other trademarks are property of their respective owners.

# Introduction to participating as a Committer

The Microsoft Biology Foundation (MBF) is a language-neutral bioinformatics toolkit, built as an extension to the Microsoft .NET Framework. MBF implements the following:

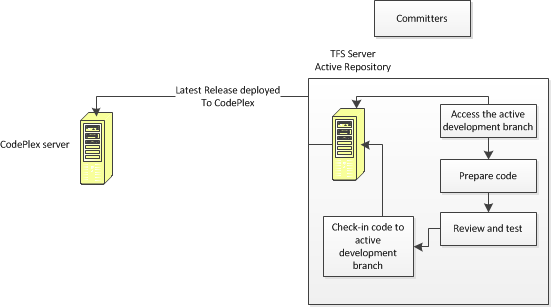
* An object model for representing genomic data.
* A range of parsers for common bioinformatics file formats.
* A range of algorithms for manipulating DNA, RNA, and protein sequences.
* A set of connectors to biological Web services such as NCBI BLAST.

MBF is available under an open source license. There are two avenues of participation in the project:

* As a Contributor – accessing the codebase deployed on Codeplex as self-contained .zip file and submitting your work through CodePlex.
* As a Committer – directly accessing the codebase in the active development repository and directly checking in your changes to the repository.

For details on the different levels of participation see the **Contribution Roles** section in the “Overview” document at <http://mbf.codeplex.com/documentation> and in TFS at ..\MBI\Doc.

Contributors and committers both have access to executables, source code, demo applications, and documentation but committers have direct access to the latest code in the active source control repository for the project development branch and can submit changes directly to the project through Visual Studio and Team Foundation Server (TFS).



You can obtain these privileges by becoming a committer. A committer has write access to the root of the project, and works on major code contributions. To become a committer you must be granted committer rights by the MBI coordinators as described in the “Becoming a committer” document at <http://mbf.codeplex.com/documentation>. If you do not already have already committer rights for MBF you may wish to become a committer.

## About committer code contributions

MBF is open to code contributions from the community, with the goal of extending the range of available functionality to researchers and life scientists everywhere.

We encourage developers who extend MBF to contribute their code back to the project as open source, so that the community as a whole can benefit from their work. Microsoft researchers are already using MBF in their research, as are an increasing number of academic partners. These researchers will be extending the range and power of MBF, and we encourage you to do the same as a committer utilizing the active development branch in TFS.

## Become a Committer and Partner

As a consistent contributor to MBF you may wish to become a committer to facilitate and enhance your participation in the project coding effort.

Besides being granted committer rights by the project coordinators, committers must also have partner credentials, which grant access to the extranet servers that host the repository. If you have partner credentials from previous MBI TFS projects, you should already have access to the extranet servers. Otherwise, you can obtain partner credentials by contacting the project team through the [MBI Community Forum](http://getsatisfaction.com/mbi) or the [MBF CodePlex Web site](http://mbf.codeplex.com/documentation).

For details on the process see the “Becoming a Committer**”** document at <http://mbf.codeplex.com/documentation>.

## Committer code review responsibilities

A committer also agrees to accept assignments to review other committer shelvesets and contributor patch submissions.

The time for review varies, depending upon where the project team is in its schedule. If the team is close to a milestone release, then your contribution might be postponed until the next milestone release. The general guideline for processing a submittal is:

1. Committers get automatic notification of contributor CodePlex patch submittals via email.
2. Committers assign Patch adoption to a specific committer within 48hrs, at which point the assigned committer contacts the contributor via CodePlex to let them know we are working on it. They may also request additional information such as unit tests, test results and test data.
3. The assigned committer responds with code review comments within72 hours of submission of all needed materials including documented code, unit tests, test results and test data.
4. If changes are required, the contributor is notified via email and the submittal process is repeated.
5. The contribution is accepted or rejected. Considerations for accepting a contributor's submission include the following:
   1. Does it meet all posted guidelines posted such as comments, coding, and testing?
   2. Does the contribution meet the quality bar?
   3. Were the unit tests included with the submission? Unit tests should be bundled up along with the new code.
   4. Were all test run and did they all return "green lights?"
6. If the contribution is accepted, the assigned committer merges the contributor's change into the Main development branch within a week of acceptance.
7. If the contribution is rejected, the assigned committer provides the contributor with the reasons for rejection.

**Note**: Tortoise is required to be able to read code submittals.

## Install TortoiseSVN

Contributors must use TortoiseSVN to submit their code contributions via CodePlex and TortoiseSVN must be use by committers to apply the patch for review. Therefore committers are required to install [TortoiseSVN.net](http://tortoisesvn.net/downloads.html) as part of your obligation to assist with contribution code reviews.

For more information go to the”Committer Onboarding” document.

For more information on applying a patch submitted via CodePlex see the [Apply and review a Contributor patch](#_Apply__and) section.

## The MBI committer contribution process

This document describes the process for you to follow when writing and contributing code to the project as a committer. The process consists of the following steps:

1. [Committer getting started](#_Toc288728937)
2. [Review the C# coding and documentation guidelines](#_Toc288728938)
3. [Review existing code](#_Toc288728939)
4. [Create a new work item](#_Toc288728940)
5. [Write new code](#_Toc288728941)
6. [Apply and review a Contributor patch](#_Toc288728942)
7. [Prepare a Shelveset for code review](#_Toc288728943)
8. [Request code review from committers](#_Toc288728944)
9. [Respond to code review recommendations](#_Toc288728945)
10. [Submit updated shelveset for review](#_Toc288728946)
11. [Submit the changes to the code base](#_Toc288728947)

Note:If you are a professional employee, please check with your employer for your company policy for submitting code to any open source code project**.**

# Committer getting started

The following section describes how to prepare to participate in the project as a committer.

## Follow on-boarding instructions

After you have been granted committer access and you have activated your account, you will have access to the MBI Extranet SharePoint described in the “Committer Onboarding” document at <http://mbf.codeplex.com/documentation>. Follow the guidelines in the “Committers Onboarding” document, to get started.

The Onboarding document describes how to:

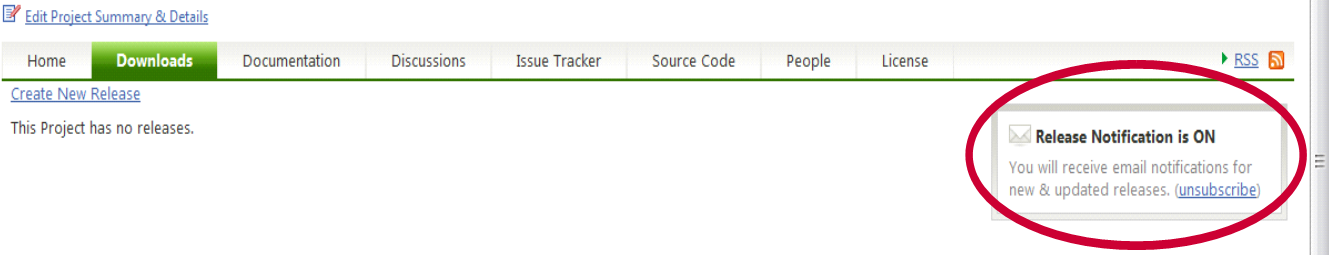
* Install the necessary tools.
* Create and configure your development environment.
* How to work with the source tree.

## Sign up for MBF email and discussion groups

The MBF community uses the CodePlex **Issue Tracker** to create and track issues such as feature work, product issues, and tasks.

You should ensure that you are signed up to receive email notifications for the following:

* Downloads: Release Notifications
* Discussions: Discussion Notifications
* Issue Tracker: Issue Notifications

  
 Setting Release Notifications Choices on CodePlex Projects

Note: You can choose to receive an update on every change or daily summaries, and you can choose RSS feeds instead of email. The important thing is to stay current with who is doing what on the project, so that you know who you might need to coordinate with.

You can review and manage your email notification settings in your CodePlex profile as well. There might be other CodePlex project notifications you will want to join. This guide focuses on the MBF groups that you should be aware of.

## Use available resources

Take advantage of the MBF resources in the following table to learn how to use MBF and to follow the community forum.

|  |  |
| --- | --- |
| Resource | Description |
| [Microsoft Biology Initiative](http://research.microsoft.com/bio) – Microsoft Research  <http://research.microsoft.com/bio> | The main MBI website. |
| [Microsoft Biology Foundation - Microsoft Research](http://research.microsoft.com/en-US/projects/bio/mbf.aspx)  [http://research.microsoft.com/bio/mbf.aspx](http://research.microsoft.com/en-us/projects/bio/mbf.aspx) | The main project website. |
| Project Training  <http://research.microsoft.com/bio/training.aspx> | Training material downloads, including hands-on labs that will help you get started coding. |
| MBI community forum <http://www.getsatisfaction.com/mbi> | A community forum where you can ask and answer questions on the project and receive updates. |

## Dependent software products

To compile the project, Visual Studio 2010 is required.

The project has the following optional components:

|  |  |
| --- | --- |
| Optional Component | Description |
| Microsoft Silverlight 3 or later [Resources | Microsoft Silverlight](http://www.microsoft.com/silverlight/resources) | Used for the MBF Sequence Assembler application |
| IronPython Runtime <http://www.codeplex.com/IronPython> | Used for the IronPython scripts |
| Trident Version 1.0 or later <http://tridentworkflow.codeplex.com> | Used for building Trident activities and workflows |
| **Sandcastle Help File Builder** <http://shfb.codeplex.com/>/ | Used to automatically generate a help file for the APIs |
| **FxCop** <http://www.microsoft.com/downloads/en/details.aspx?displaylang=en&FamilyID=917023f6-d5b7-41bb-bbc0-411a7d66cf3c> | To check for possible design, localization, performance, and security improvements in .NET managed assemblies |

For academic researchers. You can receive free Microsoft Software. To download and install the latest version of Microsoft Visual Studio® (VS2010), visit the Microsoft [DreamSpark](https://www.dreamspark.com/) web site at <https://www.dreamspark.com>.

You will also find other optional components on DreamSpark that you might like to use in your work.

# Review the C# coding and documentation guidelines

When developing code for MBF, follow the guidelines in the following documents:

|  |  |
| --- | --- |
| Document | Description |
| **C# Coding Standards** | Presents guidelines for developing code for the project. One of the key components to delivering on the MBF promise of higher productivity is the ability to provide a consistent approach to the programming model and stylistic conventions used throughout project development. |
| **How to Write Code Comments in Source Code** | Discusses how to write good documentation comments, specifically for projects that will produce an API reference directly from code comments. The quality bar is high for such projects, because there is little opportunity to modify the text after it is harvested. |
| **Testing\_Guide** | Discusses how to create and run the tests provided with MBF and any new tests for your code contribution, either as unit test or automation. |
| **Contributors Template** | Discusses how to write good documentation comments, specifically for projects that will produce an API reference directly from code comments. The quality bar is high for such projects, because there is little opportunity to modify the text after it is harvested. |

All project documentation can be found at <http://mbf.codeplex.com/documentation> and in the source tree folder $/MBI/DevelopmentBranch/MBI/Doc.

# Review existing code

As a participant in an open source project, you can evaluate the existing code first hand. Although most of the detailed technical information is available in the SDK documentation, you might find it easier to simply examine the code files and associated comments. If you use code editing software with reference/use tools such as Visual Studio, it is also helpful to traverse through the code via reference.

One other technique that we highly recommend is to build the samples provided and run them under the debugger. By setting breakpoints and stepping through actively running code, you will be able to see the order of operation is (at least for a particular sample). Also, by observing the data register during the debugging session, you can follow along with how the data is represented and modified throughout the sequence of execution.

Finally, by taking a look at what others have provided as code changes to the base framework, you can get an idea of what type of code is expected in a contribution. Changelists and ShelveSets can be quite interesting to evaluate, and will provide context for both the new and old code and the related differences.

As you become more familiar with the code and contribute to the project, you might have the opportunity to become a code reviewer, as a part of your role as a committer.

# Create a new work item

The MBI community uses TFS to create and track issues such as feature work, product issues, and tasks. As a committer you can create new work items directly in TFS. To create a work item, please see the MSDN article <http://msdn.microsoft.com/en-us/library/ms181316.aspx>.

# Write new code

At this point you have done everything necessary to begin writing and contributing code as a committer. Write your new code, following the [coding and commenting guidelines](#_Review_the_C). Build the code as described in the “[Onboarding document](#_Follow_on-boarding_instructions).” Then practice the following steps:

1. Write your new code.
2. [Write documentation for new code](#_Write_documentation_for).
3. [Write unit tests for the new code](#_Write_unit_tests).
4. Run the unit tests and confirm all tests pass.
5. [Prepare a shelveset for code review](#_Prepare_a_Shelveset).
6. [Request code review](#_Request_code_review).
7. [Submit the changes to the code base](#_Submit_changes_to).

## Write documentation for new code

Documenting the code is an integral part of the process of writing code.

If new MBF code includes new APIs for developers, then be sure to:

* Comment the APIs by following the guidelines in “How to Write Code Comments in Source Code” document at <http://mbf.codeplex.com/documentation> and in the source tree at $/MBI/DevelopmentBranch/MBI/Doc.
* The Sandcastle project is run as part of the build. Building the project will automatically produce the reference chm. For more information on using Sandcastle Help File Builder go to <http://shfb.codeplex.com/>.

**Important**: Always run **FxCop** as described in [Run FxCop](#_Run_FxCop).

## Write unit tests for the new code

Writing tests and testing code is an integral part of the process of writing code. MBF uses Visual Studio Test (VSTest) as the primary means for ensuring code quality, which makes it easy to develop, integrate, and run test code against your contributions.

You must create a new test for your code contribution, either as unit test for small changes or automation tests for new features and large changes, as described in the “Testing Guide” document at <http://mbf.codeplex.com/documentation> and in TFS at ..\MBI\Doc.

For more information on using Visual Studio 2010 test features see [Testing the Application](http://msdn.microsoft.com/en-us/library/ms182409.aspx) on MSDN.

## Run tests

The final step in producing new code is to run the tests, as described in the “Testing Guide” document. All test cases must pass before you produce and submit your shelveset.

To test new code

1. Run the new unit tests you have created and any new automation suite tests.
2. Run the tests provided with the installed project.
3. If any tests fail, fix your code and try them again.

## Run FxCop

Run **FxCop** when you finish coding. Correct any errors it finds. **FxCop** analyzes programming elements in managed assemblies by using rules that return informational messages when the rules are violated. Messages identify any relevant programming and design issues and, when it is possible, supply information about how to fix the issues.

**Note** Some errors reported by FxCop may not be consistent with the project Coding Guidelines. In general, FxCop issues should be addressed, but not if they conflict with the project guidelines. Use good judgment in applying FxCop recommendations, and it is acceptable to request new exceptions to the FxCop errors as well. The important thing to ensure is that the code you write compiles cleanly with no errors or warnings.

# Apply and review a Contributor patch

When contributors submit a patch via CodePlex it will be in the form of a TortoiseSVN DIFF file. If you have the responsibility to review that submission you must use TortoiseSVN to apply the patch to your working copy of the files.

You can select the your working copy and browse to the patch file or you can select the patch file and browse to the working copy.

**Note**: The merge program must reference the changes back to the revision against which they were made by the contributor. This is the only choice for anonymous access to the project in Codeplex. The url for accessing our repository this way is <https://mbf.svn.codeplex.com/svn/>.

To apply a patch

1. Navigate to the same folder level as was used to create the patch
   1. This can be determined from the first line of the patch file.
   2. if you pick the wrong folder level, TortoiseSVN will suggest the correct level.
2. Use the *TortoiseSVN → Create Patch* command on the parent folder.
3. From the context menu for that folder, click on **TortoiseSVN → Apply Patch**
4. In the file open dialog select the patch file to apply.
   1. By default only .patch or .diff files are shown.
   2. You can opt for “All files”.
   3. If you saved a patch to the clipboard, you can use **Open from clipboard** in the **File Open** dialog**.**
5. If the patch file has a .patch or .diff extension, right click it and select **TortoiseSVN → Apply Patch**.
6. You are prompted to enter a working copy location.
7. **TortoiseMerge** runs to merge the changes from the patch file with your working copy.
8. A window lists the files which have been changed. Double click on each file to review the changes and save the merged files.

The contributor’s patch has now been applied to your working copy. You can review and run tests normally before deciding whether or not to accept the code and check it in.

For more information go to [Creating and Applying Patches](http://tortoisesvn.net/docs/release/TortoiseSVN_en/tsvn-dug-patch.html) using TortoiseSVN on Tortoise.net.

# Prepare a Shelveset for code review

Use the Shelve command in TFS to store your pending changes—together with pending check-in notes, a comment, and a list of associated work items—on the Team Foundation Server without actually checking them into the source control server. Make sure you include all unit tests with your shelveset.

The output of the Shelve command is known as a shelveset.

For more information, see [Shelve Command in the VS Team System online help](http://msdn.microsoft.com/en-us/library/w6y8ezzs(VS.80).aspx).

**Note**: Be sure when creating a shelveset that you do not accidentally check in the changes. The Shelveset can be named, and together with your user id, a reviewer can connect to the TFS server and find your shelveset for review.

# Request code review from committers

When you are ready, you can request a code review from another project committer.

To request a code review

* Send a code review request e-mail to [msrerbio@microsoft.com](mailto:msrerbio@microsoft.com)

Include the name of your shelveset in the Subject line and in the body of your email.

The time required for a response from committers varies, depending upon where the project team is in its schedule. If the team is close to a milestone release, then your contribution might be postponed until the next milestone release.

# Respond to code review recommendations

Implement any recommendations from the reviewer. Run your unit tests again. Produce a new shelveset.

# Submit updated shelveset for review

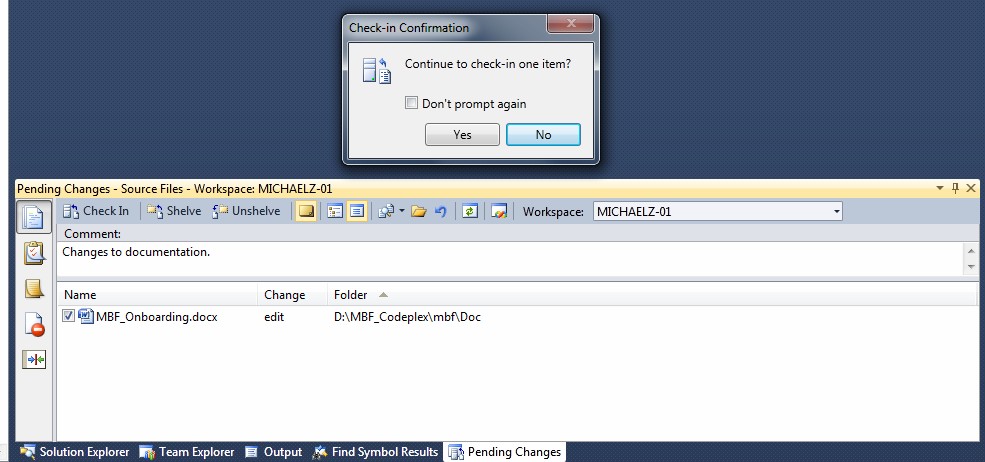
After you have reconciled recommendations for your contribution, request a review of the new shelveset and repeat if necessary.

# Submit the changes to the code base

After your work has passed review, check in your changes using the code submission feature in Visual Studio.

To check in your changes

* Click Check In in Visual Studio.



Congratulations! You have done your part to improve the project!