|  |  |
| --- | --- |
|  | Microsoft Biology Foundation Code Contribution Guide  Version 1.0 - June 2010 |

Abstract

The Microsoft® Biology Foundation (MBF) is a language-neutral bioinformatics toolkit, built as an extension to the Microsoft .NET Framework. MBF is available under an open source license.

This document describes how to contribute code to the Microsoft Biology Foundation.

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

For updates to MBF, see <http://mbf.codeplex.com>

Contents

[Introduction to the MBF contribution process 3](#_Toc264444091)

[Register at CodePlex and accept Terms of Contribution 4](#_Toc264444092)

[Review the C# coding and documentation guidelines 5](#_Toc264444093)

[Review existing code 6](#_Toc264444094)

[Contact MBF coordinators for contribution rights 6](#_Toc264444095)

[Follow onboarding instructions 7](#_Toc264444096)

[Create a new CodePlex work item 7](#_Toc264444097)

[Write the new code 8](#_Toc264444098)

[Write documentation for the code 8](#_Toc264444099)

[Write unit tests for the new code 8](#_Toc264444100)

[Prepare a Shelveset for code review 10](#_Toc264444101)

[Request code review from MBF coordinators 10](#_Toc264444102)

[Respond to code review recommendations 11](#_Toc264444103)

[Submit updated shelveset for final review 11](#_Toc264444104)

[Submit changes to the code base 11](#_Toc264444105)

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

Introduction to the MBF contribution process

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

* 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. Executables, source code, demo applications, and documentation are freely downloadable from [http//mbf.codeplex.com](file:///C:\Users\michaelz\AppData\Local\Microsoft\Windows\Temporary%20Internet%20Files\Content.Outlook\WYRO2NCW\http\mbf.codeplex.com).

## About MBF 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. If you need functions that are not in the basic library, you can implement them easily, in a way that works with the existing functions.

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 and commercial partners. These researchers will be making code contributions to extend the range and power of MBF, and we encourage you to do the same.

## The MBF contribution process

This document describes the process for you to follow when contributing code to MBF. The process consists of the following steps:

1. [Register at CodePlex and accept Terms of Contribution](#_Toc256777139)
2. [Review the C# coding and documentation guidelines](#_Toc256777140)
3. [Review existing code](#_Toc256777141)
4. [Contact MBF coordinators for contribution rights](#_Toc256777142)
5. [Follow on-boarding instructions](#_Toc256777143)
6. [Create a new CodePlex work item](#_Toc256777144)
7. [Write the new code](#_Toc256777145)
8. [Write documentation for the code](#_Toc256777146)
9. [Write unit tests for the new code](#_Toc256777147)
10. [Prepare a Shelveset for code review](#_Toc256777148)
11. [Request code review from MBF coordinators](#_Toc256777149)
12. [Respond to code review recommendations](#_Toc256777150)
13. [Submit updated shelveset for final review](#_Toc256777151)
14. [Submit changes to the code base](#_Toc256777152)

Note: If you are a Microsoft employee, any work you do on MBF must follow the Moonlighting policy in the Microsoft Employee Handbook.

# Register at CodePlex and accept Terms of Contribution

You must first sign up for an account on CodePlex. Once registered, you can create projects, enter work items, and participate in discussion forums.

To register on CodePlex

* Follow the instructions at <https://www.codeplex.com/site/register>
* Once you are registered, your first stop is <http://mbf.codeplex.com>

## Accept Terms of Contribution on the MBF home page

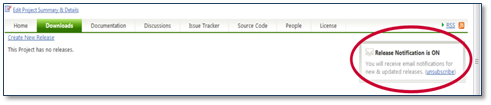
The MBF home page provides an introduction and the Terms of Contribution, which you must agree to before submitting code to the project.

Important: Please read this information carefully to ensure you understand the terms of the agreement.

## Sign up for MBF email and discussion groups

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 on top of 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.

## Dependent software products

Important: To compile the MBF project, Microsoft Visual Studio® 2010 Beta 2 or later is required.

For all contributors. In order to write, compile and execute the test components, you will need to install the NUnit framework, <http://www.nunit.org>. This is another free, open source project which the MBF framework is dependent on to maintain quality. Currently, MBF is dependent on NUnit 2.5.3.

Other optional components:

* For the IronPython scripts:  
  IronPython 2.0.1 Runtime  
  <http://www.codeplex.com/IronPython>
* For building Trident activities and workflows:  
  Trident Version 1.0 or later  
  <http://research.microsoft.com/en-us/downloads/f8d37ecb-dfed-4a3d-840a-7d1ccc6b60d4/default.aspx>
* To automatically generate a help file for the APIs:   
  **Sandcastle**  
  <http://www.sandcastledocs.com>.
* To check for possible design, localization, performance, and security improvements in .NET managed assemblies:  
  **FxCop**  
  http://msdn.microsoft.com/en-us/library/bb429476(VS.80).aspx

For academic researchers. You can receive free Microsoft software to assist in working with MBF. 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:

“MBF C# Coding Standards”

Presents guidelines for developing code on and for MBF. 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 MBF 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.

“MBF Contributors Template”

Provides a template for a User/Programming Guide for MBF contributions. The format is flexible, so you should feel free to adapt this template to the particular needs of your contribution.

All MBF documentation can be found at <http://mbf.codeplex.com/documentation>

# Review existing code

As 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 (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, in addition to your role as a contributor.

# Contact MBF coordinators for contribution rights

To add code to MBF, you must be granted contribution rights by the MBF coordinators.

To obtain contribution rights

* Send an e-mail requesting these rights to [msrerbio@microsoft.com](mailto:msrerbio@microsoft.com)

Include these details:

Your level of interest in contributing to MBF.

Your background and/or experience in programming and bioinformatics.

Your acceptance of the MBF Terms of Contribution, found at [http://mbf.codeplex.com](http://mbf.codeplex.com/) and listed here:

|  |
| --- |
| Terms of Contribution You may contribute to the Microsoft Biology Foundation (“MBF”) project only if you agree to the terms below. If you do not agree, do not contribute.  License  MBF is licensed under the Ms-PL. Any code submitted to the MBF project must be submitted under the Ms-PL license. Under no circumstances will code be accepted under any other license terms. Code submitted must be the original work of the author.  Copyright  Copyright attribution should be included in any contribution, but existing copyright attribution should not be modified in any way.  Contributions should include a header file referencing that it is licensed under the Ms-PL for code management purposes.  Existing header files with attribution can be modified to include additional attribution where appropriate.  You must add the terms of the MS-PL license to any file with new or modified copyrights.  Copyright attribution will in no way affect the terms of the license under which the code is distributed.  Code Standards  Strict adherence to the coding guidelines, as outlined on the CodePlex documentation section, should be maintained. The latest version of this document will always be available at <http://mbf.codeplex.com/documentation> and available to all users and potential contributors to the project. The purpose of this is to ensure consistency in readability as well as quality throughout the code, making it easier for others to read and comprehend as well as maintaining a high quality bar. |

# Follow onboarding instructions

After you are granted Contribution access, you can get started with MBF by following the guidelines in “Microsoft Biology Foundation Onboarding Guide,” found at <http://mbf.codeplex.com/documentation>.

The Onboarding document describes how to:

* Get ‘wired’ into the MBF team by joining discussion lists and email notifications, locating relevant web sites, and browsing the documentation.
* Install the necessary tools.
* Create and configure your development environment.

# Create a new CodePlex work item

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

For general help on the Issue Tracker, see [the CodePlex Information and Discussion wiki](http://codeplex.codeplex.com/wikipage?title=Issue%20Tracker&referringTitle=CodePlex%20Documentation).

To create a work item

1. Sign in to CodePlex and go to the MBF project.

2. Click Issue Tracker.

3. Click Create New Item.



4. Set Status to Proposed.

5. Set Type to Task.

6. Enter a clear title and description for your work item.

7. Fill out the remainder of the form as appropriate and click Save.

# Write the new code

Write your new MBF code, following the [MBF coding and commenting guidelines](#_Review_the_C). Build the code as described in the [Microsoft Biology Foundation Onboarding Guide](#_Follow_on-boarding_instructions).

Important: When you finish coding, run FxCop. 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 that some errors reported by FxCop may not be consistent with the MBF Coding Guidelines. In general, FxCop issues should be addressed, but not if they conflict with MBF 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.

# Write documentation for the code

If your MBF code includes new features for users, then fill out the MBF Contributors Template, found at <http://mbf.codeplex.com/documentation>.

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.
* Use the Sandcastle tool to automatically generate a help file for the APIs.

# Write unit tests for the new code

All code contributions top MBF require unit testing with the NUnit Framework ([www.nunit.org](http://www.nunit.org)) as described in the [MBF Onboarding document](#_Follow_on-boarding_instructions). You must create a new NUnit test for your code contribution, either as unit test or automation.

Here are some recommendations for creating new unit tests:

* Identify all the public APIs—methods, constructor, properties, or others—available in the framework.
* Create a “happy path” (the expected execution path) test case for each public API and property.
* Write test cases to cover a specific scenario, where one or more public APIs are used.
* Cover as many scenarios as possible.
* Make sure that every line of code is hit to get the maximum code coverage.
* Automate the test cases based on the priority— for example, BVTs, Priority 1 and Priority 2.

## Unit test case generation

namespace MBF.Test | MBF.Test.dll

* Identify all the public APIs—methods, constructor, properties, or others—available in the framework.
* Create a “happy path” test case for each public API and property.
* Write test cases to cover a specific scenario, where one or more public APIs are involved.

For example, the FastA parser have many different override public APIs for parsing. MBF includes many individual public API test cases, such as TestFastaWhenParsingOneOfMany().

An example scenario would be to pass an input FastA file to the parser and get the output sequence object. Then, pass those output sequence objects into FastA formatter, where it produces an output file, which is exactly similar to the original file.

Minimal negative cases are written as part of unit test cases. For example, BadContent() is a negative FastA parser test case, to check bad content in FastA file format.

## Automation test case generation

namespace MBF.TestAutomation | MBF.TestAutomation.dll

* Identify a work item/feature for automation.
* Identify all the public APIs—methods, constructor, properties. and others—available in that feature.
* Come up with all Positive, Negative and Boundary test cases and document the same in the Product Studio/Excel work sheet.
* Identify the priority for all the test cases—for example, BVT, Priority 1 and Priority 2.

As an example, consider the FastA Parser API, which uses a Parse(file-path) method exposed as public. We would test these scenarios for that API as follows:

1. Parse FastA file that has one line sequence
2. Parse FastA file of size less than 35 K (small size)
3. Parse FastA file of size more than 35 K and less than 100 K (medium size)
4. Parse FastA file of size more than 100 K and less than 350 K (large size)
5. Parse FastA file of size more than 350 K (very large size)
6. Parse FastA file that contains only DNA sequence
7. Parse FastA file that contains only RNA sequence
8. Parse FastA file that contains only Protein sequence
9. Parse FastA file that contains DNA and RNA sequence
10. Parse FastA file that contains DNA and Protein sequence
11. Parse FastA file that contains Protein and RNA sequence
12. Parse FastA file that contains DNA, RNA and Protein sequence
13. Parse FastA file that contains Ambiguous DNA sequence
14. Parse FastA file that contains Ambiguous RNA sequence
15. Parse FastA file that contains Ambiguous Protein sequence
16. Parse FastA file with valid file path
17. Parse FastA file with invalid file path
18. Parse FastA file with space in file path
19. Parse FastA file with Unicode characters in file path
20. Parse FastA file with Unicode characters in the FastA file
21. Parse FastA file with file in network location
22. Parse GenBank file instead of FastA
23. Parse GFF file instead of FastA
24. Parse SAM file instead of FastA
25. Parse FastA file that is empty
26. Parse FastA file that is too big for the parser to handle

# 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.

The output of the Shelve command is known as a shelveset. Give the shelveset a descriptive name and update your CodePlex work item with the name of the 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 Shelve set that you don’t accidentially 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. There is no need to submit code nor email any files for review.

# Request code review from MBF coordinators

When you are ready, you can request a code review from the MBF coordinators.

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, in addition to your Codeplex user name.

The time required for a response from the MBF coordinators varies, depending upon where the MBF project team is in its schedule. If the MBF 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 MBF coordinators.

# Submit updated shelveset for final review

After you have reconciles recommendations for your contribution, you can request a final review.

To request a final review

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

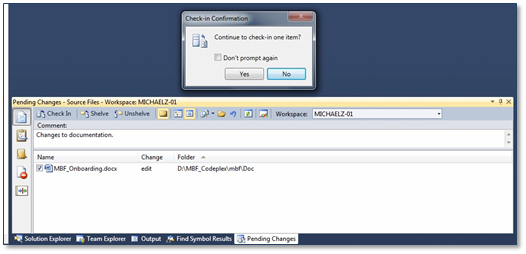
Include the name of your shelveset in the Subject line and in the body of your email.

# Submit changes to the code base

After you receive approval from the MBF coordinators, check in your changes using the code submission feature in Visual Studio.

To check in your changes

* Click Check In in Visual Studio.



Then, celebrate for having made the world a better place!