



Integral Information Solutions GmbH
Schickardstr 32 • D-71034 • Böblingen • Germany

FusionDebug 2.0

User Manual

Doc. Rev. 242, 14 February 2007

Trademarks and Warranties

FusionDebug, the FusionDebug atom, the FusionDebug Red Bug insignia and the Intergral logotype are trademarks of Intergral Information Solutions GmbH and may not be used without permission.

ColdFusion, Flex and JRun are trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Other trademarks are the property of their respective owners.

The FusionDebug software product is commercial software and may not be redistributed except with the express written agreement of Intergral Information Solutions GmbH. The software may only be used in accordance with the appropriate FusionDebug license agreement.

To the fullest extent applicable by law, Intergral Information Solutions hereby disclaims all warranties, including but not limited to the implied warranties of merchantability and fitness for a particular purpose.

Feedback

We welcome feedback on all our products and publications. To discover the various ways of contacting us, please go to the following location: <http://www.fusion-reactor.com/contact.html>. We will address your feedback as quickly as possible.

Published in Germany

Copyright © 2005-2007 Intergral Information Solutions GmbH

All Rights Reserved

2.0.0/20070213/Rev. A



Message from the Team

We'd like to thank you for using FusionDebug, the interactive debugger for ColdFusion. We're very excited to be able to bring you this groundbreaking product!

We believe FusionDebug defines the state of the art for ColdFusion debugging and provides a very important tool in the current ColdFusion development world. With the advent of ColdFusion-driven Flex applications and ColdFusion's own powerful object-oriented features, an interactive debugger has become essential for effective and productive debugging. We believe you will be able to use FusionDebug to accelerate your development and improve quality across the whole software cycle.

If you're coming to us from Dreamweaver, CFEclipse or even a text editor, don't worry. You don't have to use FusionDebug or Eclipse to edit pages, or change your development platform. You can still develop with Dreamweaver or whatever your favorite editor happens to be, and jump into FusionDebug whenever you need to work out a problem. FusionDebug is able to use your files from whatever software you used to create them.

Writing an interactive debugger is no small undertaking, and for a scripting language like ColdFusion it's been a remarkable and rewarding challenge. We want you to be aware that there are some limitations due mostly to the way ColdFusion compiles and optimizes pages prior to running them. When stepping through code, you may occasionally see the current line jump back to the start of a tag, or move to a location you didn't expect. In these cases, FusionDebug will of course continue to display accurate variable and expression data.

We will of course continue to improve FusionDebug and find solutions to such issues in future releases, but we believe they have only a minimal impact when compared to the huge benefits and productivity gains FusionDebug delivers.

Finally, we'd like to ask you for your support. We want to make FusionDebug as good as it possibly can be and you can play the most important role to help us to help everyone. Please provide us with feedback on problems, features and ideas at the following location: <http://www.fusion-reactor.com/contact.html>; we truly appreciate it.

Thank you!

The FusionDebug Team



Table of Contents

1 What's New in 2.0	9
2 Getting Started	11
3 Introduction & Concepts	13
3.1 Why Do I Need a ColdFusion Debugger?.....	13
3.2 What Do I Need to Get started?.....	13
4 Debugger Concepts: Quick Tour	15
4.1 Debug Configuration.....	15
4.2 Debug Target.....	16
4.3 Threads.....	16
4.4 Stack Frames and the Stack.....	16
4.5 Variables.....	17
4.6 Expressions.....	17
4.7 Breakpoints and the Current Instruction Pointer.....	18
4.8 Stepping.....	18
4.8.1 Step Over	18
4.8.2 Step Into	19
4.8.3 Step Return	19
4.8.4 Resume	19
4.9 Terminate	19
4.10 Source Code Lookup.....	20
5 Installing FusionDebug	21
5.1 Using the Complete Installer.....	21
5.2 Installing the Eclipse Platform.....	21
5.2.1 Downloading and Running Eclipse.....	21
5.3 Views and Perspectives.....	22
5.4 Installing the FusionDebug Plugin.....	24
5.4.1 Installation via the Eclipse Software Install/Update Manager.....	24
5.4.2 Installation via the Standalone Installer.....	25
5.4.3 The Server Configuration Wizard.....	26
6 Manually Setting up Java for Debugging	29
6.1 Setting up ColdFusion for Debugging.....	29
7 Introducing the FusionDebug Perspective	31
8 Configuring a Project in Eclipse	33
8.1 Creating a new Eclipse Project.....	33



8.2 Creating a FusionDebug Configuration.....	33
9 Exploring FusionDebug.....	37
10 Installing and Activating a License.....	39
10.1 Installing a License.....	39
10.2 Activating a License Automatically.....	39
10.3 Activating a License Manually.....	39
11 The Source Lookup Tab.....	41
11.1 What is the Source Code Lookup Tab?.....	41
11.2 What is a Lookup?.....	42
11.3 How Do I Find the Source Code Lookup Tab?	43
11.4 How do I Use the Source Code Lookup Tab?.....	44
11.5 Example Configurations.....	45
11.5.1 Single Machine Setup.....	45
11.5.2 Remote Debugging Setup.....	45
11.5.3 Multiple Project Setup.....	46
11.5.4 CF Mappings Setup.....	47
11.5.5 Alternative Webserver Setup	47
11.5.6 Linux and UNIX Setup	48
11.6 Why Are My Breakpoints Not Active?.....	49
11.7 Why Is FusionDebug Unable to Display My Source Code?.....	49
12 Reference.....	51
12.1 Debug View.....	51
12.2 Breakpoints View.....	53
12.3 Expressions View.....	54
12.4 Eclipse Editor Context Menu.....	55
12.5 Configuration Dialog.....	57
12.5.1 Connect Tab.....	58
12.5.2 Source Code Lookup Tab.....	59
12.5.3 Common Tab.....	60
13 Troubleshooting.....	61



Table of Figures

Figure 1: The Default Debug Configuration in Eclipse.....	15
Figure 2: Debug Target.....	16
Figure 3: Stack With Two Frames.....	16
Figure 4: Variables View.....	17
Figure 5: Expressions View.....	17
Figure 6: Breakpoint and the Current Instruction Pointer (CIP).....	18
Figure 7: Debug View with Stepping Tools Highlighted.....	18
Figure 8: Eclipse Welcome Screen.....	21
Figure 9: Eclipse Java Perspective.....	22
Figure 10: Eclipse Install/Update Dialog.....	24
Figure 11: Eclipse Update Search Results Dialog.....	25
Figure 12: Installer Select Components Screen.....	26
Figure 13: Configuration Wizard jvm.config Select.....	27
Figure 14: FusionDebug Perspective.....	31
Figure 15: Setting the 'All Projects' Mapping.....	34
Figure 16: FusionDebug Configuration for Local Machine.....	35
Figure 17: Successful Connection.....	35
Figure 18: How Lookups are Used.....	41
Figure 19: How Lookups are Used.....	42
Figure 20: Setting Lookups for the Example.....	42
Figure 21: The Run -> Debug Menu.....	43
Figure 22: The Eclipse Toolbar Debug Dropdown.....	43
Figure 23: Creating a New Debug Configuration.....	43
Figure 24: The Source Code Lookup Tab.....	44
Figure 25: The Source Code Lookup Add/Modify Panel.....	44
Figure 26: The Source Code Lookup Rules Panel.....	44
Figure 27: Single Machine Setup.....	45
Figure 28: Remote Debugging Setup.....	45
Figure 29: Multiple Project Setup.....	46
Figure 30: CF Mappings Setup.....	47
Figure 31: IIS Lookup.....	47
Figure 32: Apache Lookup.....	48
Figure 33: Virtual Directory Lookup.....	48
Figure 34: UNIX Apache Lookup.....	48



Figure 35: Setting up for UNIX Debugging.....48

Figure 36: Breakpoint Not Active..... 49

Figure 37: Source Code Not Found Editor..... 49

Figure 38: The Debug Drop-Down Menu..... 57

1 What's New in 2.0

We're excited to be able to bring you FusionDebug 2.0, and we'd like to thank our users for the many great suggestions and bug reports you sent us – we truly appreciate it! Here's a quick rundown of what's new in 2.0.

Source Code Lookups

We have extensively refined how FusionDebug looks up source code. We also now provide immediate feedback in the **Breakpoints View** about exactly **which** file **on the server** contains the breakpoint, or whether a server file could not be found.

We've completely redesigned the **Mappings** tab of the configuration dialog (which is now called **Source Code Lookups**) to make it easy to add lookups for all projects, and for specific Eclipse folders.

Where necessary, we've extended this user manual with appropriate sections which introduce the concept of Source Code Lookups ("Source Code Lookup", on page 20, and a full explanation "The Source Lookup Tab" on page 41).

This should result in far fewer cases of breakpoints not firing, and fewer cases of FusionDebug asking you to find a file manually.

Debug Perspective

We've added a **FusionDebug Perspective**, which collects together all the views you need to begin debugging ColdFusion applications. This removes the need to customize the Debug Perspective, and should get you up and running with FusionDebug faster.

Installer

In addition to the standard **Eclipse Software Install/Update Manager** package which can be installed directly within Eclipse from our website, we're also now providing a traditional **installer**, which will be more familiar for many users.

Server Configuration Wizard

We're also providing a **Server Configuration Wizard** which is capable of updating your ColdFusion/JRun server configuration file in order to allow ColdFusion to accept connections from FusionDebug. This should make setting up your configuration easier; of course, you're still free to edit this file by hand (and we provide comprehensive instructions on how to do this).

Inspect Expressions

Instead of creating watch expressions, it's now possible to highlight an expression in your code and have it immediately evaluated and displayed in a dialog box.

Icon changes

We've unified our use of the Red Bug icon (you can see him at the bottom of this page) across the board: in dialogs, menus and documentation. This should make identifying FusionDebug components within Eclipse much easier.

Bug Fixes

We've also worked hard to eliminate the bugs which you reported to us from our 1.0.0 release.



2 Getting Started

Before we get into the manual, you might like to know that we have several Captivate videos on line at the following URL:

<http://www.fusion-reactor.com/fusiondebug/gettingStarted.html>

These will show you how to set breakpoints, use stepping, watch expressions and examine variables. You can download the files used in the videos from:

<http://www.fusion-reactor.com/fusiondebug/helpDocs/FusionDebugExamples.zip>

For more details on any of the subjects covered in these videos, please refer back to this manual.



3 Introduction & Concepts

Welcome to **FusionDebug**, the interactive debugger for ColdFusion applications. This document introduces the concepts you'll need to start debugging your own applications, guides you through the configuration process and gives you some helpful hints on using FusionDebug to best effect.

FusionDebug is delivered as stand-alone install package, or as a plugin for the **Eclipse Platform**. Eclipse is a universal framework for applications – not just debuggers – and provides many features that make developing interactive applications simpler. We selected Eclipse because it already has a good tradition of being a comprehensive, well-supported, stable platform, and a proven architecture for debuggers.

Many other plugins are available for Eclipse – **CFEclipse**, for example, is a leading environment for writing ColdFusion scripts. If you have CFEclipse installed too, FusionDebug will integrate with that environment when it needs to open an editor. If not, don't worry – FusionDebug will also use the standard Eclipse text editor.

Please note that, as our recommended CF editing plugin, we give you the option of automatically installing CFEclipse directly from our stand-alone install package.

3.1 Why Do I Need a ColdFusion Debugger?

Debugging ColdFusion pages usually requires using lots of `<cfoutput>` statements, `<cfdump>` or writing to log files. But as applications grow more sophisticated with the addition of object oriented components, and the integration of Flex, it's becoming increasingly difficult to solve complex problems.

FusionDebug solves these problems by allowing you to suspend any page or component **during its execution** and see all the variables used by that page. You can add specific **watch expressions**, you can even **set variable** values while the page is suspended. Pages can then be single-stepped or resumed.

You may think you don't need a ColdFusion debugger because you've never used one.

FusionDebug will change the way you work.

3.2 What Do I Need to Get started?

The easiest way to get started is by downloading our complete installer. This will let you install everything you need to get developing: **Eclipse**, **CFEclipse** and **FusionDebug**.

If you are already using Eclipse or Flex Builder 2, we have a second installer which you can use to automatically add these plugins into your existing environment.

Additionally, both installers give you the option of running the **Server Configuration Tool** which will automatically update your server settings to enable debugging.

If you do use the complete installer, once you run FusionDebug, you will see a short set of simple instructions which will get you debugging in minutes.

To download FusionDebug or find out about other ways to install, visit our website at:

<http://www.fusiondebug.com/>



4 Debugger Concepts: Quick Tour

We know that not everyone's used a debugger before, so we're providing this section to introduce all the required concepts to get you started. We've put in some screenshots to show you what each item looks like. Don't worry about finding these things in Eclipse yet – we'll cover that in later sections.

4.1 Debug Configuration

A **Debug Configuration** is a collection of parameters which allow FusionDebug to find, connect to and use a **Debug Target** – a ColdFusion instance – as well as enabling FusionDebug to look up your source code whenever it needs to.

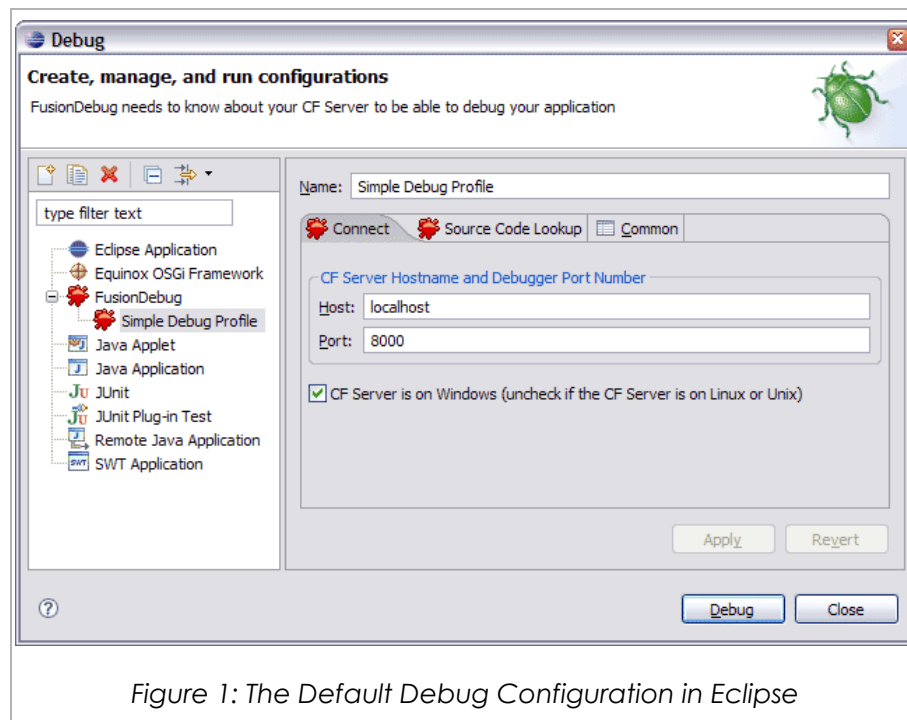


Figure 1: The Default Debug Configuration in Eclipse

If you have installed FusionDebug as a complete package (including Eclipse), you will already have a **Simple Debug Profile** defined. Simply follow the steps listed in the default document in order to complete this profile and start debugging.



4.2 Debug Target

The **Debug Target** is the JRun/ColdFusion instance to which you want to connect to perform debugging. The Debug Target is specified in the Debug Configuration and appears in the Debug View. In Figure 2, you can see the Debug Configuration (Local Configuration) and the Debug Target (`localhost:8000`).

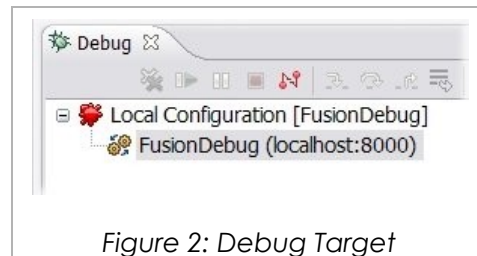


Figure 2: Debug Target

4.3 Threads

A **Thread** is a 'worker unit' that runs ColdFusion pages. Threads are idle most of the time, sitting and waiting for web server connections. When a page is requested, JRun/ColdFusion picks the next idle thread and gives it the request to execute. The thread's job is then to execute the script, performing any actions the script requires, then to return the generated page back to the web browser. While the thread is busy running a request, it can't handle another request.

When the thread's job is complete, it returns to being idle.

In ColdFusion, these worker threads are named **jrpp-X** or **web-X** where **X** is a number. You'll see them appear in the debugger as you learn to pause and inspect them. In Figure 3 for example, we see that the thread `jrpp-1` is handling the request.

4.4 Stack Frames and the Stack

The **Stack** is a list of pages currently being executed by a thread, along with the current line number of that page.

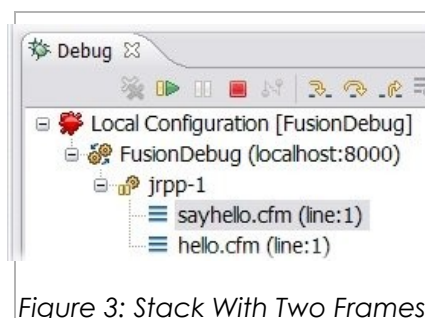


Figure 3: Stack With Two Frames

You'll see the stack in the Debug View. If you only have a single simple page, the stack will only contain one element – that page. If you have a more complex architecture, your stack may contain many elements.

For instance, if you have a simple page – `hello.cfm` – and you run this page, the stack will contain just this page.



If your page calls a sub-tag `sayhello.cfm`, the stack will contain two levels: `hello.cfm` and `sayhello.cfm`. This scenario is demonstrated in Figure 3: the `jrpp-1` thread is handling this request.

We can see that the page which ran first - `hello.cfm` - is at the deepest level - the bottom of the stack.

We call each of these levels a **Stack Frame**. The stack is composed of frames, which represent each page in the current call.

4.5 Variables

When a thread is paused, the Variables View shows you all the ColdFusion variables currently available in the selected stack frame. Using the example above, if you select `sayhello.cfm`, the Variables View would show you all ColdFusion scopes and variables available within that page. By selecting the `hello.cfm` page you could then see the variables available to that page, and so on.

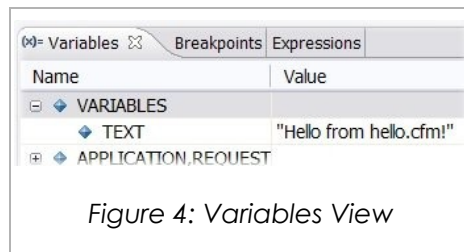
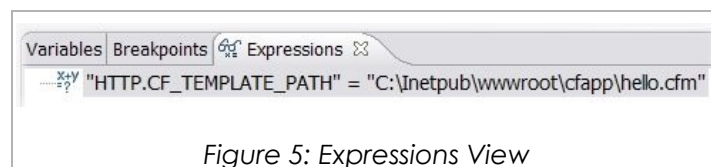


Figure 4 shows the Variables View displaying the `text` variable, which contains a friendly string.

4.6 Expressions

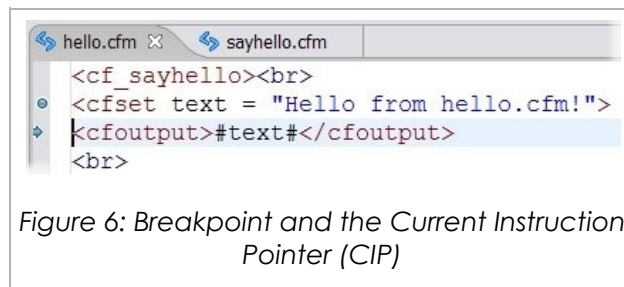
Expressions are variables you are specifically interested in watching, or expressions you refer to frequently. The Expressions View allows you to carefully inspect just the variables you're currently focused on.

The Expressions View is evaluated every time a thread pauses, so you can quickly see how the variables are changing during the execution of a page. Most ColdFusion expressions are valid in the Expressions View. Figure 5 shows an Expression View inspecting the `CF_TEMPLATE_PATH` variable inside the HTTP scope.



4.7 Breakpoints and the Current Instruction Pointer

Breakpoints are a key feature of FusionDebug, and instruct FusionDebug to pause any request whenever it hits the breakpoint. We say that the breakpoint has been **hit**, or that it has **fired**. Breakpoints appear within Eclipse editors in the left margin as small blue dots. Figure 6 shows a breakpoint set on the `<cfset>` tag. Be aware that in order for a breakpoints to fire, the associated **project must be open**. Breakpoints within closed projects are disabled.



This is the only way to cause threads to pause inside FusionDebug. Once they are paused, the Variables and Expressions views come alive and allow you to examine the state of your application.

Breakpoints can be **enabled** or **disabled** (in which case they won't fire). You can remove them and reset them at any time.

All current breakpoints are displayed in the **Breakpoints View**. In addition to the exact file and line, FusionDebug also displays the full path to the file (see section 12.2, "Breakpoints View", on page 53). The full path is computed using the **Source Code Lookup** tab (see section 4.10 below, and for a full explanation, section 11, "The Source Lookup Tab", on page 41).

Current Instruction Pointer

The Current Instruction Pointer (CIP) is a small blue right-pointing arrow which appears in the left margin of the Eclipse editor. It shows you the line of code that ColdFusion is about to execute. The CIP moves when you execute one of the Stepping commands to show you exactly which line will be executed next. Figure 6 shows the CIP on the `<cfoutput>` tag, on the line immediately below the breakpoint. The CIP is useful during **Stepping** operations to help you see what ColdFusion is executing.

4.8 Stepping

Once a thread has hit a breakpoint and paused, you can perform one of four actions on it. These actions are known as **Step** actions. They're accessible using the icons at the top of the Debug View, as shown in Figure 7.



4.8.1 Step Over

Step Over simply tells FusionDebug to execute the line of code currently at the Current Instruction Pointer. No other actions are performed. If the Current Instruction Pointer is



located at the last ColdFusion statement of a page and there are no other pages in the stack, the Step Over command is equivalent to the Resume command, and the page will run to completion. If there are further pages in the stack, Step Over will execute the command and return to the next page in the stack (the caller).

4.8.2 Step Into

Step Into is useful when the Current Instruction Pointer is located at a ColdFusion tag which would cause a sub-tag or CFC to be executed. If you execute a **Step Into** command while on such a tag, FusionDebug will attempt to locate the source code for that page (or CFC) and step into it, pausing on the first line of that page.

If the Current Instruction Pointer is not located on a sub-tag or CFC call, this command is equivalent to the **Step Over** command.

4.8.3 Step Return

Step Return is only available from within sub-tags and CFC methods, and causes FusionDebug to immediately finish executing the sub-tag or method, and return to the calling page, where execution will pause ready for further Step commands.

4.8.4 Resume

Resume causes FusionDebug to start execution of the page once again, until any further breakpoints are hit, or the page completes normally.

4.9 Terminate

If a thread is suspended, the **Terminate** command causes it to be stopped completely. This command can be used by selecting either a stack frame or a thread – in either case it is the thread itself which is terminated.

When terminated, no further execution will take place in the thread, including all pages, CFCs and objects associated with that request, and nothing further will be returned to the browser. This might be useful to cancel a thread which is about to do something destructive, or begin a time-consuming task.

Caveats

There are a couple of points to be aware of with the **Terminate** command.

- To be as reliable as possible, FusionDebug uses a very harsh method of halting the thread. This stops the request exactly at the point at which it's currently suspended, and cancels all further execution. If you have database transactions pending, they should be rolled back automatically. However, **Terminate** itself does not guarantee this.
- Terminate may not be effective during the following types of operations:
 - CF/CFC pages using Java objects via the `<cfobject>` tag, which are executing native methods (used for things like file and socket operations)
 - CF/CFC pages performing file operations themselves
 - CF/CFC pages waiting for data from a socket (e.g. gateway pages etc.)

In these situations, Terminate may either be delayed until these operations complete, or may not be effective at all. However since this command is only available when a thread is stopped, these situations should only arise rarely.



4.10 Source Code Lookup

When you set a breakpoint within a ColdFusion source file, FusionDebug must translate the breakpoint information into a form that ColdFusion can understand.

To do this, FusionDebug must know exactly where the page is on the server.

When a breakpoint fires, or a stepping action occurs, the reverse must also happen: FusionDebug must translate the debugging data from the server into a form FusionDebug can use.

Pages can be addressed in many different ways (e.g. Web services, CFCs and subtags), and the **same name** may occur many times on your server (e.g. **Application.cfm**). When dealing with this name, FusionDebug needs an unambiguous way of telling which page is which.

The first way FusionDebug differentiates different pages is using the **Eclipse Folder Structure**, but that's not always enough: some projects have the same structure – framework-based projects especially.

So in order to provide another level of checking, FusionDebug needs to know **where** on your server these pages are and to **which** Eclipse project these paths are associated.

The **Source Code Lookup** tab allows you to enter one or more **Lookups**. These **lookups** enable FusionDebug to accurately find files when you set a breakpoint, and when it needs to open a source file, for example during a **Step Into** operation.

The **Source Code Lookup** system is explained in much more detail later, in section 11, "The Source Lookup Tab" on page 41.



5 Installing FusionDebug

This section provides guidance on getting FusionDebug up and running on a simple project. You should follow the steps in order and learn how to control FusionDebug on this small project before using it on your own.

The files for this project are available here:

<http://www.fusion-reactor.com/fusiondebug/helpDocs/FusionDebugExamples.zip>

5.1 Using the Complete Installer

The **complete installer** from our website supplies **Eclipse** and **FusionDebug**, together with optional component **CFEclipse**.

If you wish to install Eclipse separately or you wish to install FusionDebug into Flex Builder 2 then please read the next sections.

5.2 Installing the Eclipse Platform

If you already have Eclipse 3.1 or 3.2 installed – perhaps because you have Macromedia Flex 2 Builder installed already – and are comfortable with Perspectives and Views, you can skip ahead to section 5.4 - "Installing the FusionDebug Plugin", on page 24.

5.2.1 Downloading and Running Eclipse

If you have not already installed Eclipse, you will need to download and install it from the Eclipse website:

<http://www.eclipse.org/downloads/>

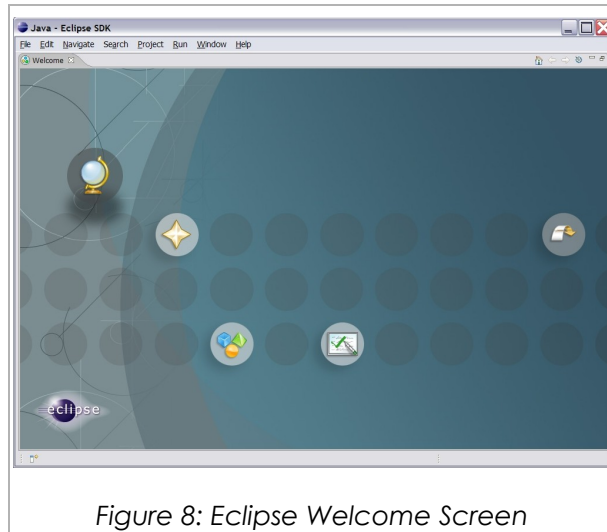
Be sure to select the **Eclipse SDK**. At the time of going to press, Eclipse 3.2.1 is the current revision. We have tested FusionDebug thoroughly with Eclipse 3.2.1 and recommend this platform for FusionDebug.

Eclipse is provided as a zip (compressed) file. When the download has completed, you should uncompress this file to a folder on your hard disk.

Once uncompressed, Eclipse can be started by double-clicking the `eclipse.exe` program (Windows) or running the `eclipse` script from a terminal window (Unix). When Eclipse starts up for the first time, it will display the Welcome screen, as shown in Figure 8.

You may like to place a shortcut to Eclipse on your desktop to make it easier to launch in future.





After launching for the first time, Eclipse will ask where it should locate your **Workspace**. This is an Eclipse storage area where **new** projects will be created by default, and where Eclipse will store its settings. Accept the default, and optionally check the box marked "Use this as the default and do not ask again".

Don't worry if you have existing ColdFusion projects on your disk – Eclipse will be able to use those too, even if they are not located in the workspace. You can also change the location of the workspace later. You may take some of the Eclipse tours by selecting one of the round icons.

Proceed directly to the workbench by selecting the **Workbench** icon (the rightmost round button) or the white 'X' on the Welcome tab.

5.3 Views and Perspectives

Eclipse is now installed and ready for use, looking something like Figure 9. This is the **Java Perspective**. You can see the currently-active perspective at the top-right of the window.



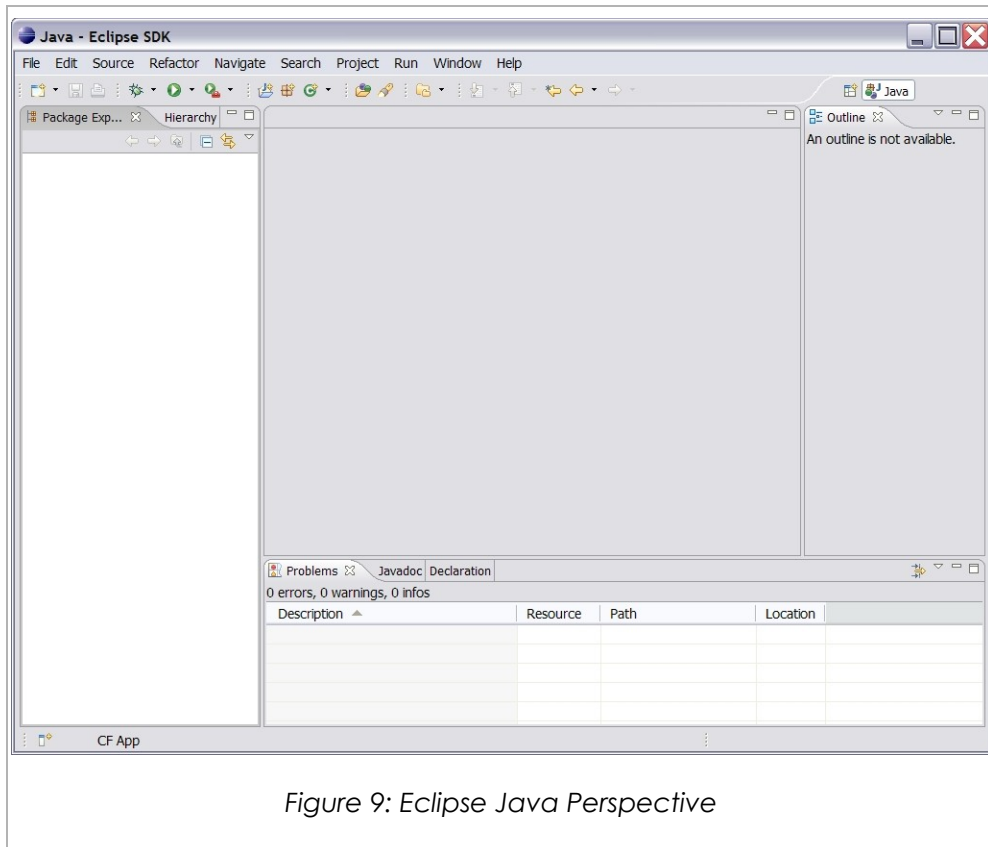


Figure 9: Eclipse Java Perspective

An Eclipse Perspective is a collection of related Views. You can see that this perspective contains (amongst others) the **Package Explorer** view on the left, and the **Outline View** on the right.

Views can easily be dragged into different locations.

If you accidentally close a view, it can be reopened again by selecting **Windows -> Show View**. If it doesn't appear in that menu, selecting **Other** will allow you to find the view in the complete list.

After installing FusionDebug, the **FusionDebug Perspective** will be available. This contains all the views you will need to debug ColdFusion code.

5.4 Installing the FusionDebug Plugin

The FusionDebug Plugin is supplied in two different ways:

- **As an on-line Eclipse Software Install/Update Manager package**
This package is installed entirely within Eclipse using the **Eclipse Software Install/Update Manager**. If you are familiar with Eclipse, or don't need the additional configuration supplied by the installer (see below) use this method.
- **As a standalone installer**
This package is supplied as a standard install executable, like most other software. It is able to locate an existing Eclipse installation and install or update the FusionDebug located there. The installer also supplies the **FusionDebug Server Configuration Wizard**, which can modify the **jvm.config** file of an existing ColdFusion installation in order to allow it to accept debugging connections. If you are not familiar with the Eclipse Software Install/Update manager, or you'd like the installer to update your ColdFusion configuration for debugging, use this method.

5.4.1 Installation via the Eclipse Software Install/Update Manager

Once Eclipse is installed and configured, we are ready to install FusionDebug using the Eclipse Update Manager. This will download, check and install the FusionDebug package, and makes upgrading FusionDebug easier in the future.

- In Eclipse, select **Help -> Software Updates -> Find and Install** to begin the process. This activates the **Install/Update** dialog, shown in Figure 10.

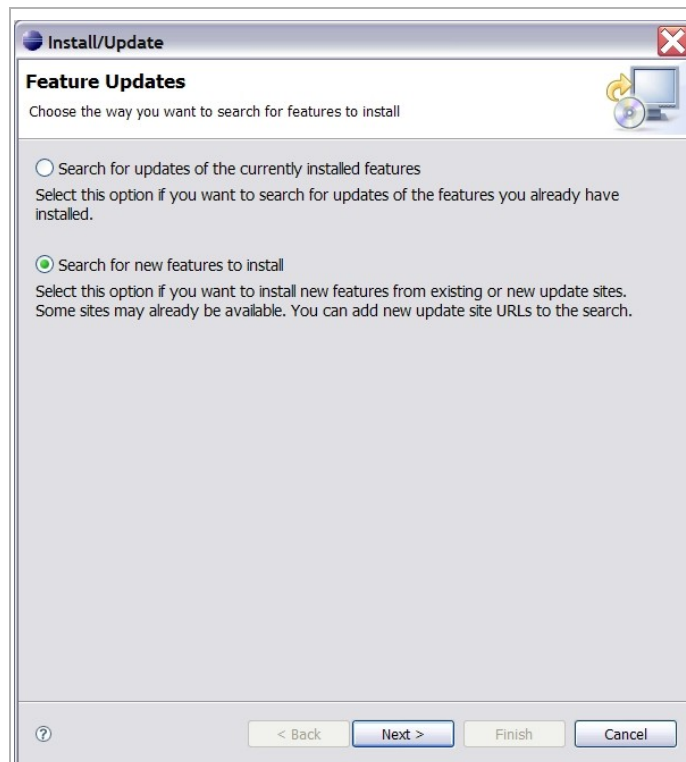


Figure 10: Eclipse Install/Update Dialog

- Select **Search for new features to install**, and click **Next**.



- Select **New Remote Site** to add the Intergral FusionDebug software site. In the dialog which appears, enter
 - Name: **FusionDebug Update Site**
 - URL: <http://www.fusion-reactor.com/fusiondebug/update>
- ... then click **OK**.
- In the Install/Update dialog, ensure the **FusionDebug Update Site** is checked, then click **Finish**.
- The Update Manager will connect to **fusion-reactor.com** and locate the new software. A search dialog will appear, similar to that pictured in Figure 11. Place a check mark against the topmost tree element ("FusionDebug Update Site") and click **Next**.

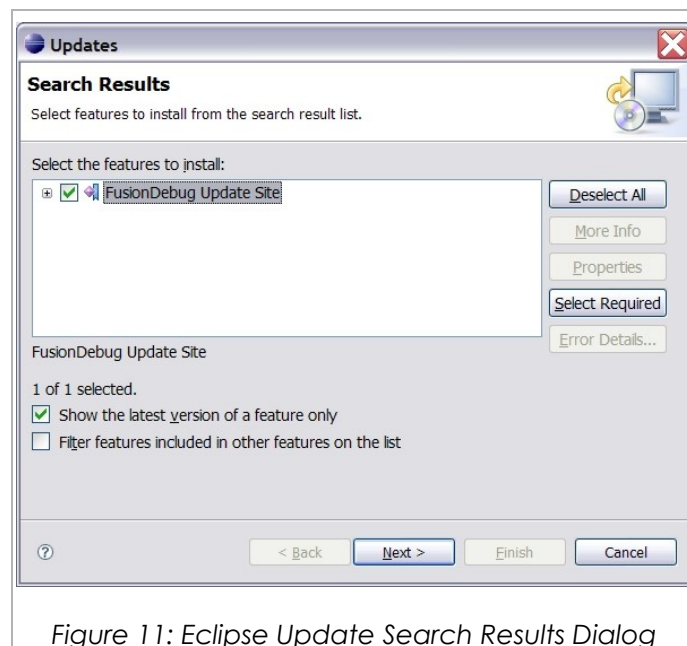


Figure 11: Eclipse Update Search Results Dialog

- Read the license agreement which appears in the next step, and if you accept its terms, select **I accept the terms in the license agreement** and click **Next**.
- Finally, click **Finish** to complete the process. The update manager will download and install the package automatically (about 14 MB).
- When the **Feature Verification** dialog appears, click **Install All**.
- Eclipse will ask if it should restart the Workbench. Answer **Yes** to this question. The workbench will restart.

5.4.2 Installation via the Standalone Installer

Instead of using the **Install/Update Manager**, you may prefer to use the **Standalone Installer**. This installer will add the FusionDebug plugin into an existing Eclipse installation. You also have the option of installing the **Server Configuration Wizard** which can be used to automatically configure a local copy of ColdFusion for debugging.

The Installer can also be used to upgrade existing FusionDebug installations, even if they were previously installed using one of the other install methods.

The installer has the following steps:



- The Server Configuration Wizard initially displays its welcome page. Click **Next**.
- The **Destination Directory** dialog will appear. Enter the install path for FusionDebug. Note that this is **not** the Eclipse path where FusionDebug will be installed: it is the location where the **FusionDebug User Manual, Server Configuration Wizard** and **Uninstaller** will be installed. Click **Next**.
- The installer now displays the **Select Components** dialog which can be seen in Figure 12. The installer can be used to update a copy of Eclipse, install the Server Configuration Wizard, or both. Once you have selected the components you wish to install, click **Next**.

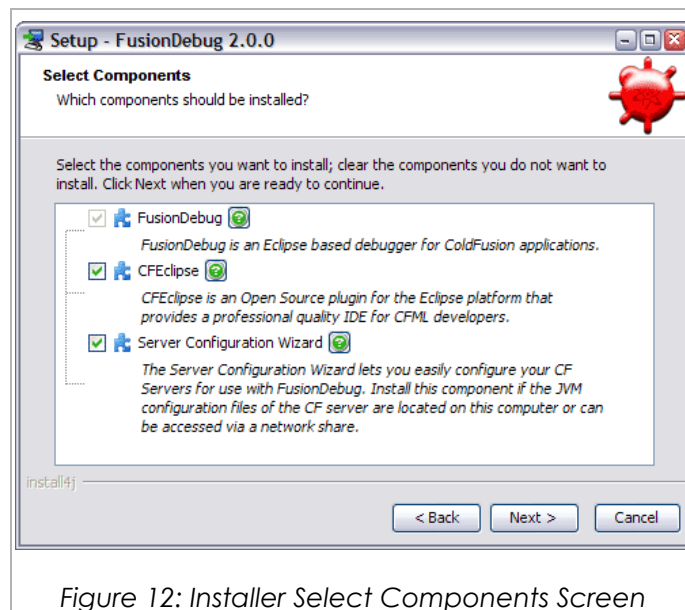


Figure 12: Installer Select Components Screen

- The **Select Eclipse dialog** will appear next. If you opted to install the Eclipse plugin from the previous screen, then you should now enter to the path to your Eclipse installation. On Windows the path will typically be something like "C:\Eclipse" or "C:\Flex Builder 2". Click **Next**.
- The **Start Menu Folder** dialog will now appear. Select where you want FusionDebug to appear in your Start Menu, or select not to have FusionDebug added to your Start Menu if you prefer. Click **Next**.
- The **Additional Tasks** dialog appears next. Check the boxes displayed to specify if the installer should create **Desktop** and **Quick Launch icons**. Click **Next**.
- FusionDebug is now installed. If you opted to also install the **Server Configuration Wizard** then you will have the option to run it immediately. Click **Finish**.

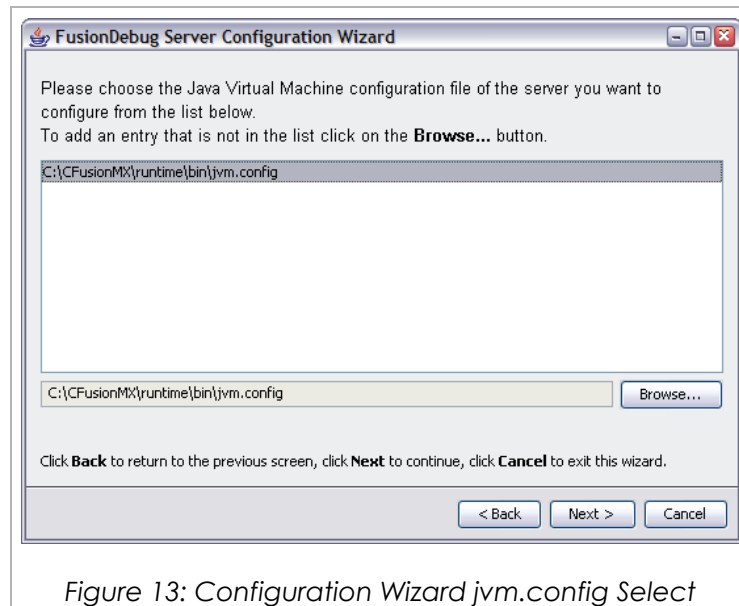
5.4.3 The Server Configuration Wizard

If you choose to run the **Server Configuration Wizard** directly from the installer, or if you start it later from the Start Menu, it will guide you through the following steps:

- Initially you will see a **Welcome** dialog which will explain what the wizard intends to do. Click **Next**.
- The Server Configuration Wizard will attempt to find any local ColdFusion servers on your computer (see Figure 13). Simply select the one you want to enable for debugging. If your ColdFusion installation is not listed then use the **Browse** button to



locate the `jvm.config` file associated with your ColdFusion server. (To make the necessary changes, the Server Configuration Wizard must have **write access** to this file.) Click **Next**.



- The **Port Select** dialog lets you pick which port the Debugger will run on. Select a free port and **note down this number as you will need it when setting up a debugging profile within Eclipse**. Click **Next**.
- The Server Configuration Wizard will now update your CF configuration. Once completed, click **Next**.
- The final page lets you know that a backup of your `jvm.config` file has been written and where it can be found. Typically they will be stored under your ColdFusion install directory under `/runtime/bin/jvm.config.bak`. Click **Finish** to close the wizard.



6 Manually Setting up Java for Debugging

If you have already used the **FusionDebug Server Configuration Wizard** to set up your ColdFusion environment for debugging, you may safely **skip this step** and go directly to section 7, "Introducing the FusionDebug Perspective" on page 31.

This section illustrates how to configure ColdFusion to allow FusionDebug connections. We will use JRun as our example, as this is a very common ColdFusion platform.

6.1 Setting up ColdFusion for Debugging

This procedure adds the required debugging parameters to a standard ColdFusion (JRun) installation.

1. Stop all running ColdFusion services.
2. Locate the Java configuration file `jvm.config`¹.
 - This can be found in the following location:
 - Windows: `C:\CfusionMX7\runtime\bin\`
 - Unix standalone: `/opt/cfusionmx7/bin/`
 - Unix multi-instance: `/opt/jrun4/bin/`
3. Make a copy of this file, and store the copy somewhere on your disk. In case of problems, you can restore the copy to this location.
4. Open this file in a text editor (e.g. Notepad).
5. Locate the line `java.args`. These are the settings used to start the Java Virtual Machine, in which ColdFusion runs.
6. **Remove** this option:
 - `-XX:+UseParallelGC`
7. **Add** these options:
 - `-Djava.compiler=NONE -Xnoagent -Xdebug`
`-Xrunjdpw:transport=dt_socket,server=y,suspend=n,address=8000`
8. All options must be on the same line. Be careful not to introduce any line breaks. The following line is an example of a complete `java.args` parameter line:


```
java.args = -DJINTEGRA_NATIVE_MODE -DJINTEGRA_PREFETCH_ENUMS -Xmx512m
-XX:MaxPermSize=128m
-Dsun.io.useCanonCaches=false -Dcoldfusion.rootDir={application.home}/../
-Dcoldfusion.libPath={application.home}/../lib -Djava.compiler=NONE -Xnoagent
-Xdebug -Xrunjdpw:transport=dt_socket,server=y,suspend=n,address=8000
-Dcoldfusion
.classPath={application.home}/../classes,{application.home}/../lib/updates,{applic
ation.home}/../lib/,{application.home}/../gateway/lib/,{application.home}/../wwwroot/
WEB-INF/cfform/jars
```
9. If you have other applications running on port 8000, change the port address to a free port. Make a note of the new port
10. Start all ColdFusion services.

¹ For non-JRun installations, this file may be named differently. Refer to your J2EE platform documentation for more information on changing the Java options used during startup.



For ColdFusion multi-instance installations, there can be an issue with each instance attempting to use the same `jvm.config` file, causing a port conflict as the second and subsequent instances come up. For the most up-to-date information on avoiding and working around this limitation, please check our support area at:

<http://www.fusion-reactor.com/fusiondebug/support.html>



7 Introducing the FusionDebug Perspective

After installing FusionDebug, a new Eclipse perspective is available. The **FusionDebug Perspective** is illustrated in Figure 14, and can be opened by selecting **Window -> Open Perspective -> Other** and selecting the **FusionDebug** perspective.

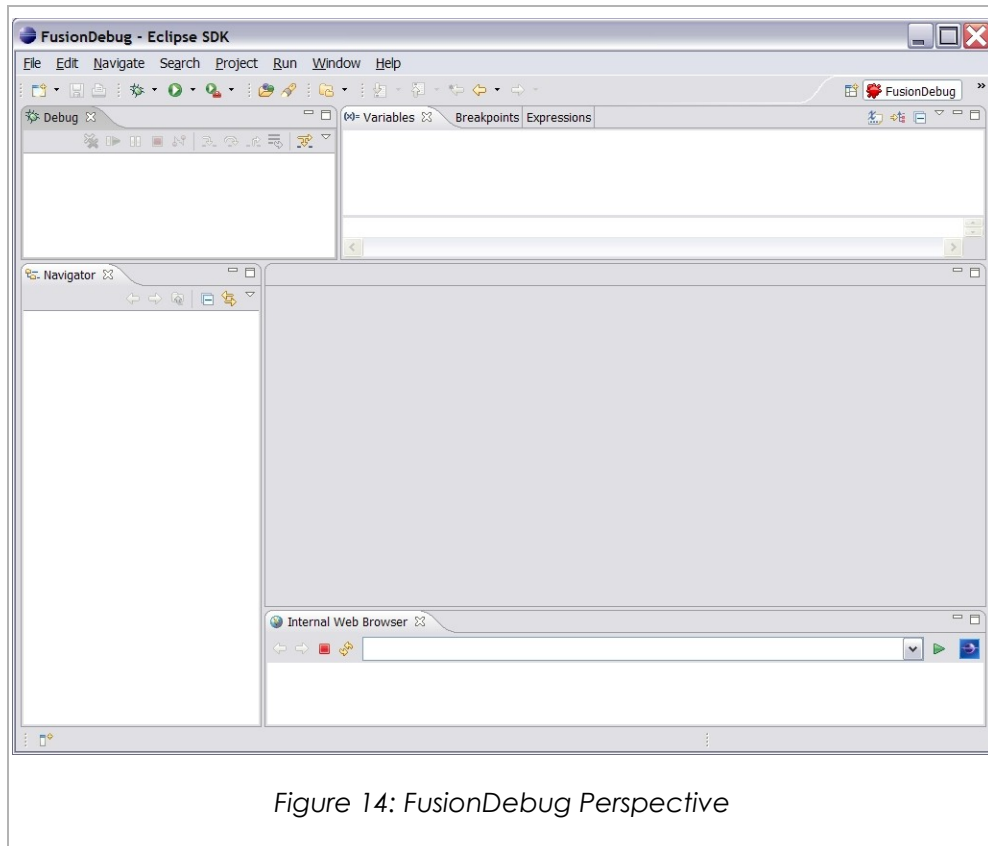


Figure 14: FusionDebug Perspective

This perspective is designed to get you started debugging ColdFusion pages. The views and their functionality are described more fully in section 12, "Reference" on page 51. Broadly, they are as follows:

- **Debug View** – shows the current **Debug Target** to which FusionDebug is connected, along with any **Stack Frames** which correspond to paused pages.
- **Variables** – when a page is suspended, this view shows all the ColdFusion **scopes** and **variables** currently available to the page.
- **Breakpoints** – lists all the **breakpoints** currently in use, together with their exact location on the server.
- **Expressions** – lists all the **expressions** currently being watched. This view is updated whenever ColdFusion pauses due to a **breakpoint** or **stepping action**.
- **Navigator** – shows all **projects** and **folders** currently in the **workspace**.
- **Editor Area** – this is the largest area in the Perspective, and is initially empty. Any editors opened will appear here.
- **Internal Web Browser** – this is a simple web browser and can be used to call pages to trigger breakpoints.



8 Configuring a Project in Eclipse

If you have installed FusionDebug using the **Complete Install** then you will automatically have a project and default debugging profile defined. In order to complete your installation, simply read the instructions shown on the initial start page.

Before Eclipse can be used to debug ColdFusion applications, you must configure a project in the Navigator. This enables FusionDebug to locate the correct source code to respond to debugging events like breakpoints, steps etc.

8.1 Creating a new Eclipse Project

In our example, we have installed some test files inside the default IIS web folder, and we're going to use these as the basis of our project.

The files are located in `c:\inetpub\wwwroot\cfapp`.

1. Start Eclipse if it's not already running, and switch to the FusionDebug Perspective if it does not open automatically (Eclipse remembers the last perspective you used before you shut it down last).
2. In the **Navigator** view, right click and select **New -> Project**.
3. The **New Project** wizard appears.
4. Open the **General** tree branch, and select **Project**.
5. Enter a meaningful name for your project (we're using "CF Test Application"), and **uncheck** the option **'Use default location'**.
 - Since we already have existing files, we can use these instead of starting a new project in the workspace.
6. Locate the folder containing the files we want to create a new project for (in `c:\inetpub\wwwroot\cfapp`) using the **Browse** button, and click **OK**.
7. Click **Finish** to complete the wizard.

After completing the wizard, the new project appears. You can double-click on any files within the project to begin editing them. Do not delete the `.project` file – that's an Eclipse internal file used to keep track of your project.

8.2 Creating a FusionDebug Configuration

This section shows you how to set up a FusionDebug configuration to connect to your local machine and begin debugging. It's very important to carefully read and understand these instructions, since setting up a working FusionDebug Configuration is a key step.

1. Open the Debug dialog using the **Run -> Debug** menu option.
2. Right-click the **FusionDebug** element on the left of the dialog and select **New**. The FusionDebug configuration page appears in the dialog.
3. Enter a meaningful name in the **Name:** field. This should be something which describes this configuration. We're using "FusionDebug Local Connection".
4. Since we're going to connect to our local machine, leave the **Host** as it is – `localhost`. The port defaults to **8000**. If you didn't change it when running the Server Configuration Wizard (see section 5.4.3), or setting up JRun for debugging (see section 6.1), leave it as it is. Otherwise, enter the same port here as you used previously.
5. Check the first option.



- **CF Server is on Windows**
This option tells FusionDebug to connect to a Windows system. Our system is Windows XP Professional, so we leave this **checked**.
6. Before we can complete the configuration, we must add at least one **Source Code Lookup** to tell FusionDebug where the ColdFusion files are located on the server. Click on the **Source Code Lookup** tab to bring up the **Lookup Table**.

This table tells FusionDebug where the files are located on the server. You must enter at least one rule in this table.

When you **launch** the configuration later, FusionDebug will compute the actual locations of the breakpoints, and display them along with the breakpoints themselves in the **Breakpoints View**. Since we only have one project, we'll use the global **<All Projects>** rule and the `c:\inetpub\wwwroot\cfapp` folder. Enter this data in the upper area and click **Add**. Figure 15 shows the Source Code Lookup tab containing the All Projects rule.

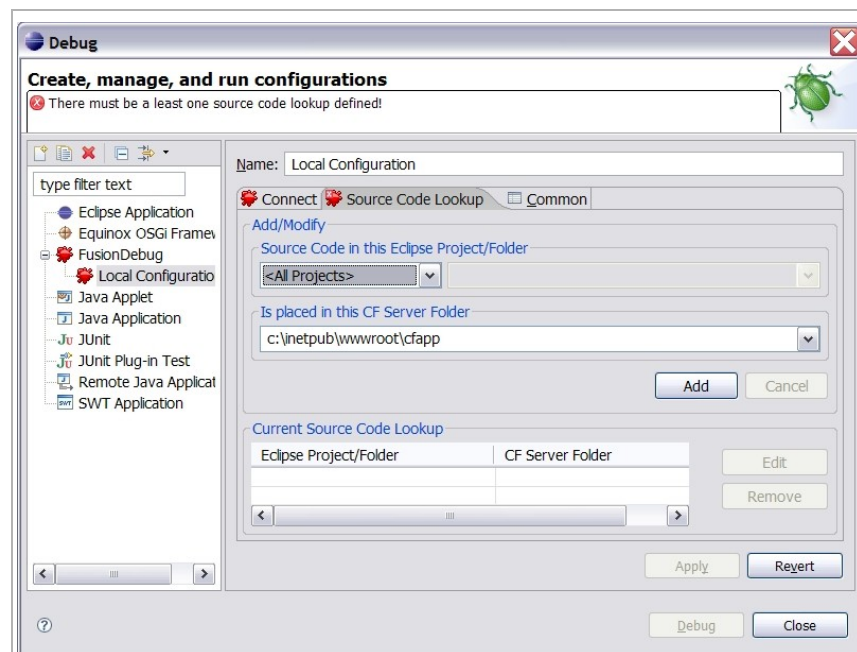
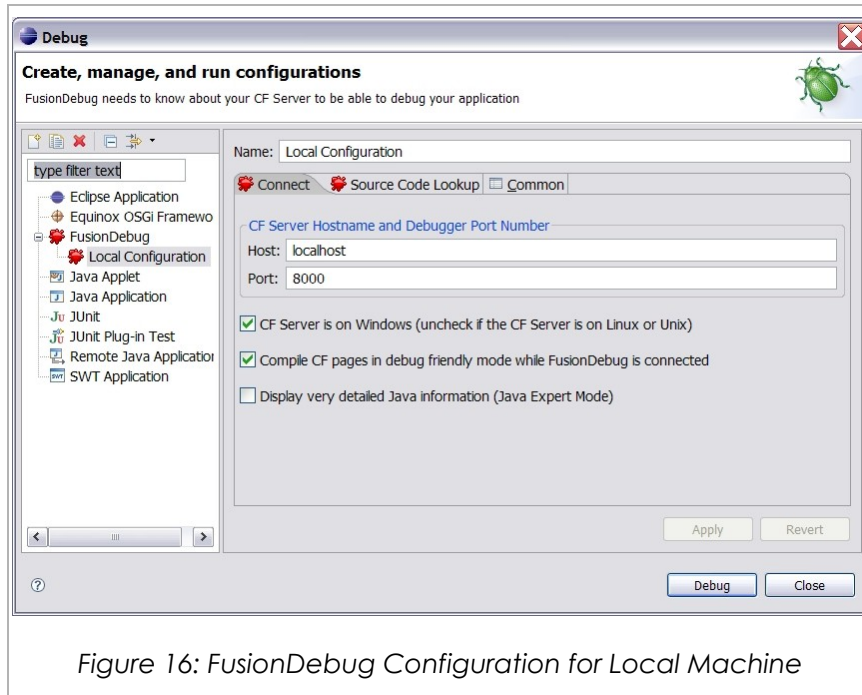


Figure 15: Setting the 'All Projects' Mapping

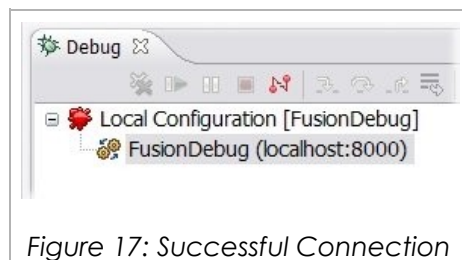
Because the **Source Code Lookup** tab is so important, we've explained it in more detail in its own section, 11, "The Source Lookup Tab" on page 41, and the exact meaning of the **Source Lookup Tab** is discussed in section 12.5, "Configuration Dialog" on page 57.



7. With the configuration looking similar to Figure 16, select **Apply**.



8. Finally, select **Debug**. The dialog should disappear, and FusionDebug should connect to the local Java instance. If this succeeds, the **Debug View** will display the debug configuration and debug target, as shown in Figure 17.



9 Exploring FusionDebug

We believe a static manual isn't the best place to demonstrate what FusionDebug can do, so we've prepared some Captivate videos instead.

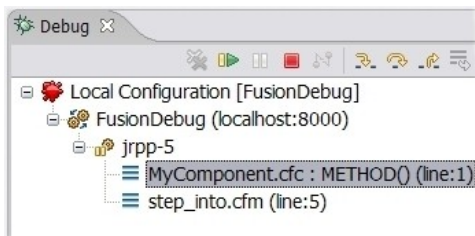
<http://www.fusion-reactor.com/fusiondebug/gettingStarted.html>

These will show you how to set breakpoints, use stepping, watch expressions and examine variables. You can download the files used in the videos from:

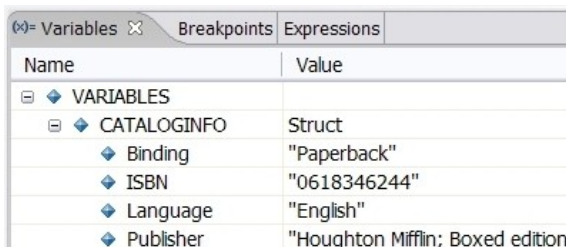
<http://www.fusion-reactor.com/fusiondebug/helpDocs/FusionDebugExamples.zip>

You should have a look through the reference sections to clear up any questions you might have, and feel free to drop us a line to the support address if there's anything you feel isn't addressed.

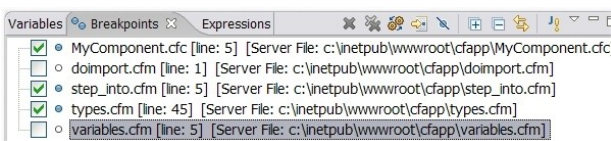
Here's a quick rundown of the main features.



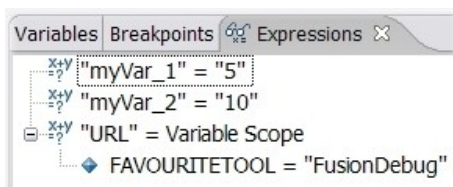
Explore Stacks... FusionDebug shows you exactly what CFCs, pages and objects are involved in your request. If you're inside a CFC function, the name of the function is also displayed.



Explore Variables... FusionDebug shows you all ColdFusion variables scopes, including handy **Current Query**, **Local Variables** and **Function Local** variables too.



Explore Breakpoints... FusionDebug keeps track of all your project breakpoints, allowing you to delete them one by one, disable selected breakpoints, or skip them altogether. The Breakpoint View also shows exactly which server file the breakpoint is in.



Explore Expressions... You can add almost any ColdFusion expression and watch the value. The values are updated in real time, whenever a thread steps or pauses, so you can see exactly what ColdFusion sees. Additionally, you can inspect any variable at any time by highlighting it in your code and selecting "Inspect Expression" from the right-click menu.



10 Installing and Activating a License

FusionDebug runs for 20 days without requiring an active license. After this period, you will not be able to configure FusionDebug or launch any debug configurations until a license is installed and activated.

To acquire a license, please visit

<http://www.fusion-reactor.com/fusiondebug/buy.html>

10.1 Installing a License

When you have received your license file, it can be installed as follows.

1. Open Eclipse and select **Window -> Preferences**.
2. Open **FusionDebug** and select the **License** node.
3. Use the **Browse** button to locate your license file, and select **Open**.
4. Finally, click **Install License**.
 - You now have 10 days to activate this license.
 - If you have web access from this PC, we recommend immediately activating the license.

10.2 Activating a License Automatically

If you have web access from the licensed PC, you can perform an automatic activation as follows.

1. Open the **License** panel as described above.
2. Click **Activate License**.

10.3 Activating a License Manually

A license can be manually activated as follows:

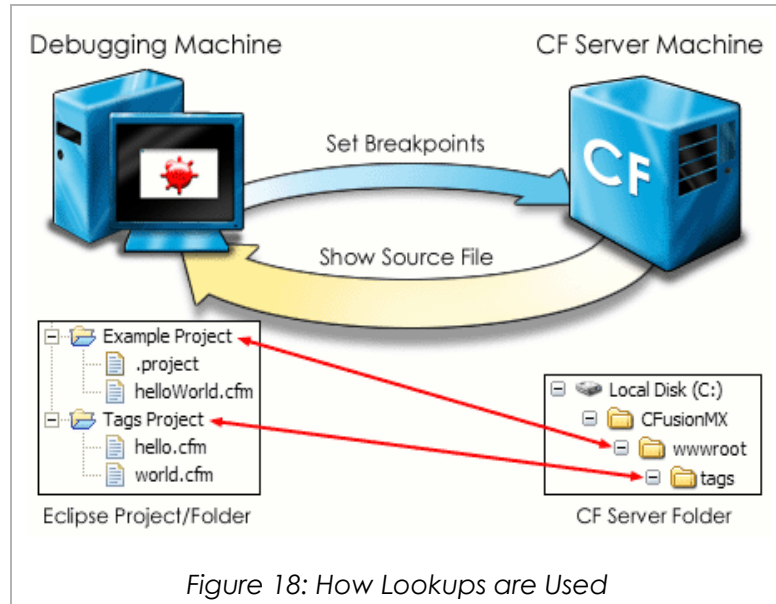
1. Open the **License** panel as described above.
2. Select the **Manual Activation** link, and note down the **Activation Input String**
 - Do not restart Eclipse until you have completed this procedure.
3. On a PC with web access, visit the **URL** mentioned in the dialog.
4. Enter the **Activation Input String** into the form and select **Submit**.
5. When the page returns, note down the **Activation Key**.
6. On the PC to be licensed, enter the **Activation Key** into the dialog box.
7. Click **OK**.



11 The Source Lookup Tab

11.1 What is the Source Code Lookup Tab?

The Source Lookup tab is used to tell FusionDebug where the source files in your Eclipse projects are placed on to your CF server.



FusionDebug uses the source code lookup rules to tell the CF server where to set breakpoints and to tell Eclipse which source code to display when a breakpoint is hit. Breakpoints are set within Eclipse but must also be transferred to the CF server for them to become active.

By configuring your Source Code Lookups, FusionDebug can support multiple Eclipse projects, mapped CFCs and include files and provide support for the various frameworks that can be used with CF.

11.2 What is a Lookup?

Often, the structure of the files in your Eclipse projects is not the same as the structure when deployed to your CF server. This leads to the problem that FusionDebug is unable to work out which files in Eclipse relate to which files on your CF server. This problem is solved by creating Source Code Lookups in your FusionDebug configuration. Each lookup consists of an Eclipse project path and the absolute path to the folder on the CF Server that contains the same files.

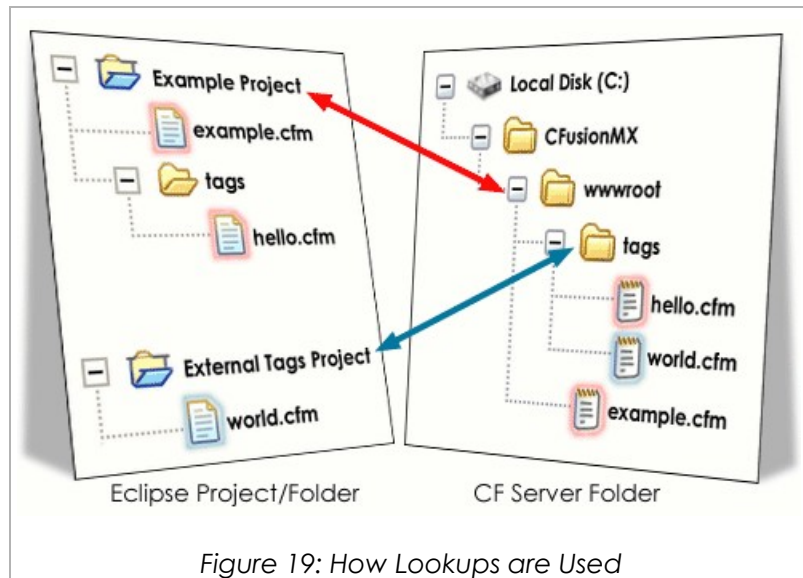


Figure 19: How Lookups are Used

If you take the Eclipse Project/Folder paths and absolute CF server paths from Figure 19, then your Source Code Lookups will look like this:

Eclipse Project/Folder	CF Server Folder
Example Project	C:\CFusionMX\wwwroot
External Tags Project	C:\CFusionMX\wwwroot\tags

Figure 20: Setting Lookups for the Example

11.3 How Do I Find the Source Code Lookup Tab?

There are two ways to find the Source Code Lookup tab. Firstly, you can open the **Run** menu and then select the **Debug...** item.

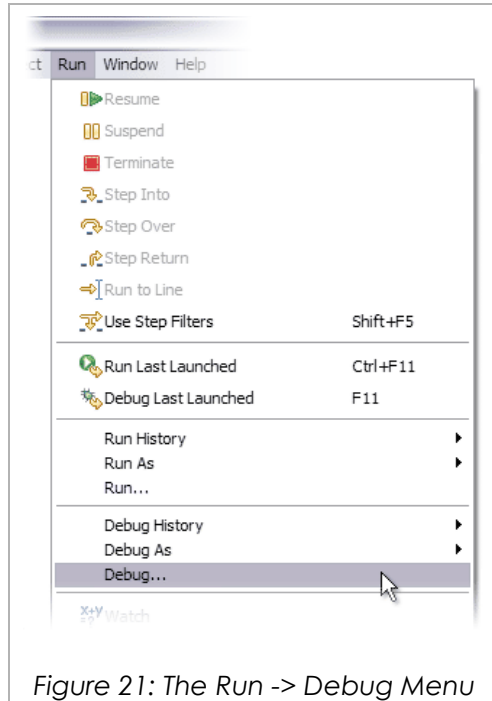


Figure 21: The Run -> Debug Menu

Alternatively, you can go to the little green debug icon in your Eclipse toolbar and click the small arrow on its right-hand side. This brings up a small menu and you can select the **Debug...** item from there.

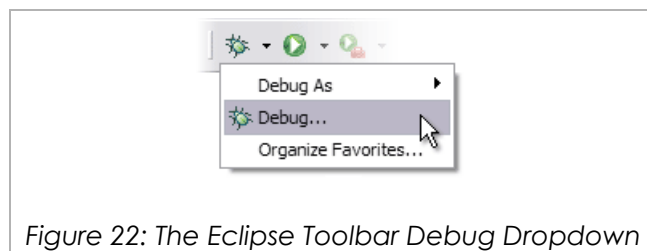


Figure 22: The Eclipse Toolbar Debug Dropdown

Once the debug dialog is visible you can bring up the **Source Code Lookup** tab by opening the FusionDebug item and clicking on an existing debug profile, or by right-clicking on the FusionDebug item and selecting **New** from the menu.

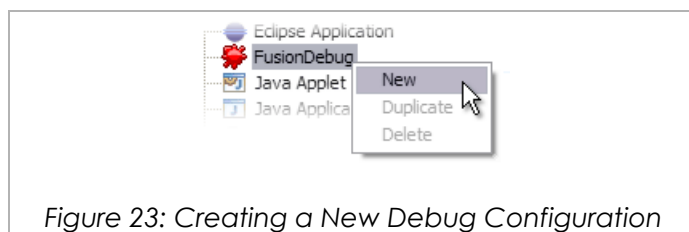


Figure 23: Creating a New Debug Configuration

11.4 How do I Use the Source Code Lookup Tab?

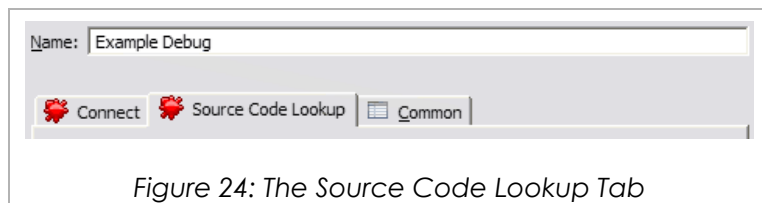


Figure 24: The Source Code Lookup Tab

The Source Code Lookup tab is split into two main sections. The top section allows you to add or modify lookups.

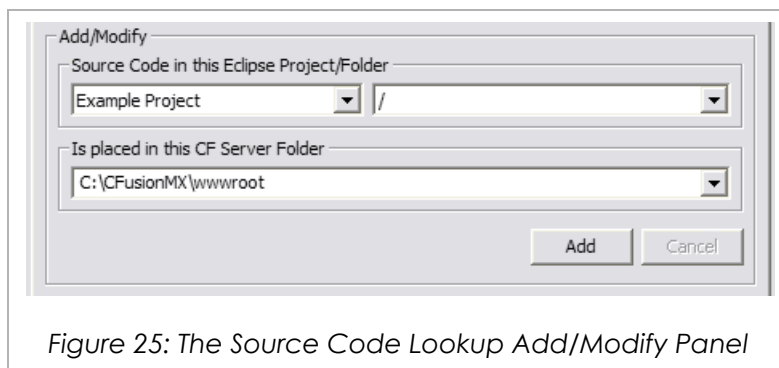


Figure 25: The Source Code Lookup Add/Modify Panel

The bottom section lists all of the lookups currently configured.

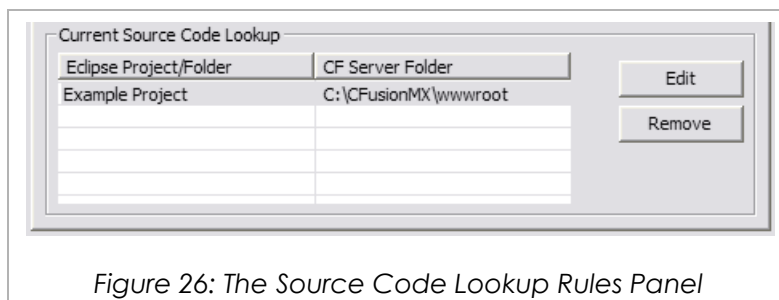


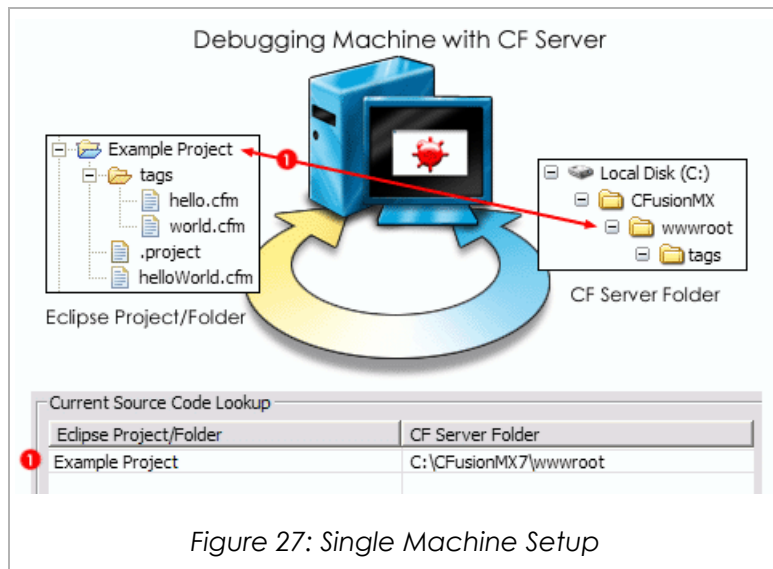
Figure 26: The Source Code Lookup Rules Panel

To add a lookup, first select an Eclipse project and optionally a folder within that project. Next, type in the file path of the folder under your CF server that you want to associate with this Eclipse path. (This is the actual path on the server itself, not the URL which you would type into a browser.)

11.5 Example Configurations

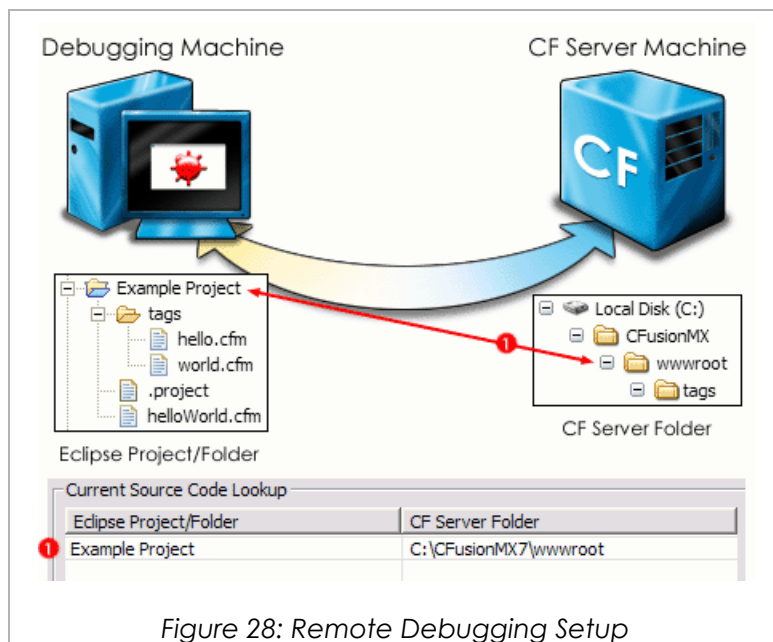
11.5.1 Single Machine Setup

The simplest configuration of FusionDebug would be if you had set up your Eclipse workspace to be the same as your CF webserver root. You would be editing the same files that are being served and you would only need a single lookup.



11.5.2 Remote Debugging Setup

If you copy your project source files directly to a remote server without changing the structure then the configuration is equally simple. You create a single lookup from your Eclipse project to the CF webserver root on that machine.



11.5.3 Multiple Project Setup

Let's say that you have an application which uses a tag library. In Eclipse that tag library is a separate project which has two folders within it. On the webserver, the contents of both of these folders end up in a single folder called "tags". You would need three lookups in this case. One from your application project to the CF webserver root and then another two from each folder in the library project to the "tags" folder.

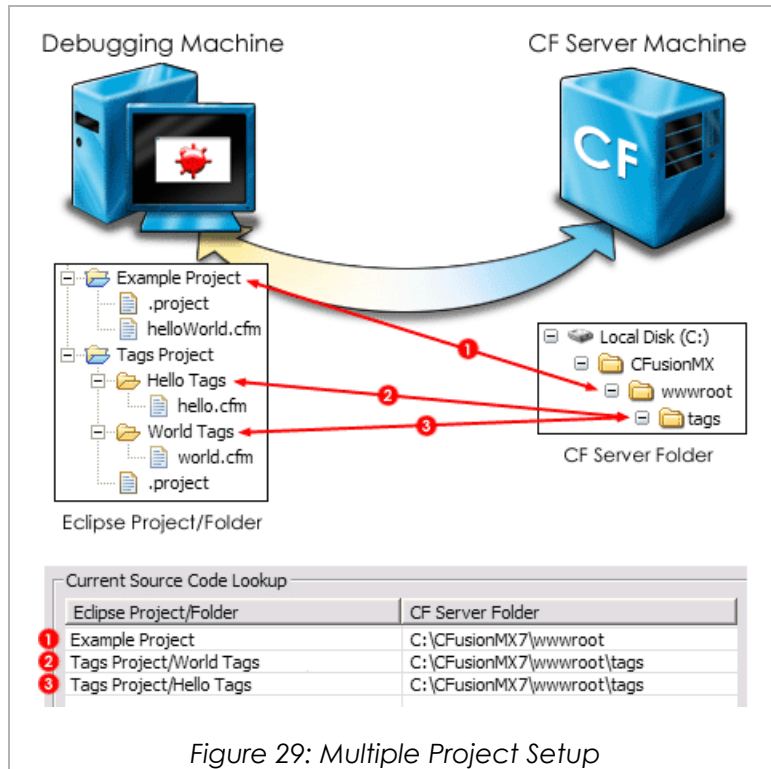


Figure 29: Multiple Project Setup

11.5.4 CF Mappings Setup

From the CF Administrator you can set up Mappings. These allow you to store cfm code in any folder on your server. They can then be referenced within your CF pages by using their logical path. To debug these files you need to add a lookup to the directory path specified in the CF mapping.

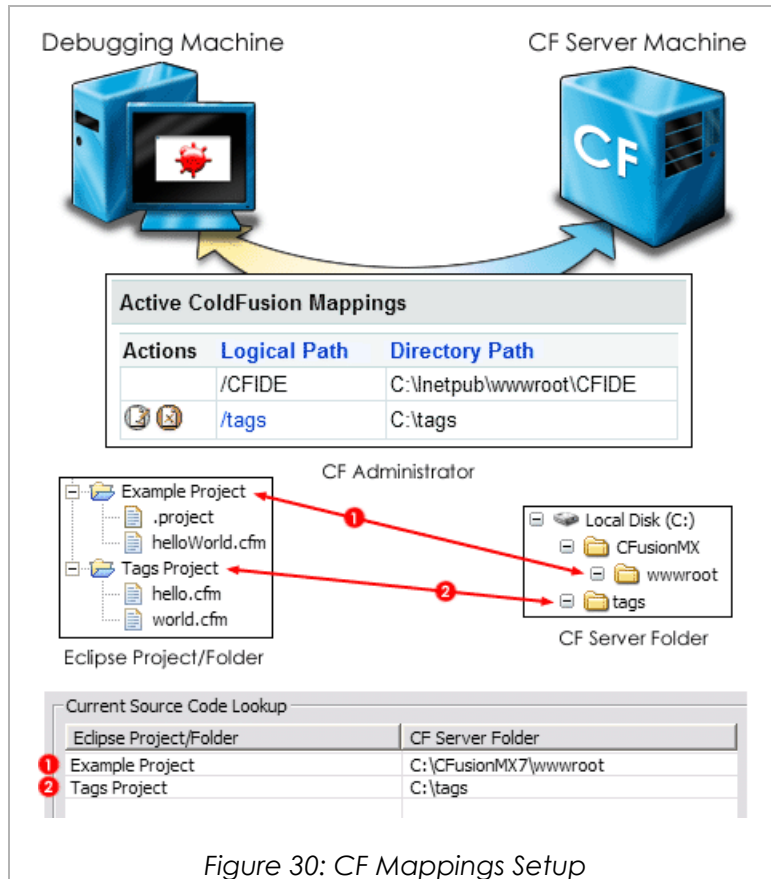


Figure 30: CF Mappings Setup

11.5.5 Alternative Webserver Setup

For simplicity, all of the examples so far have used the CF internal webserver. This doesn't mean that you can't use other web servers. Below is an example of a lookup for IIS:

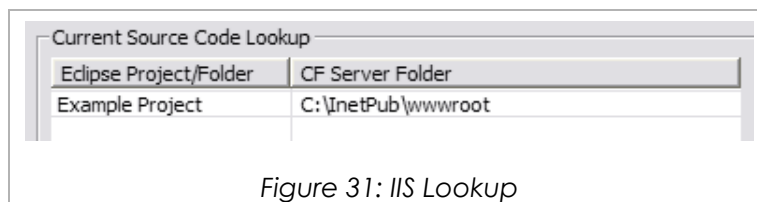


Figure 31: IIS Lookup



Below is a lookup for Apache:

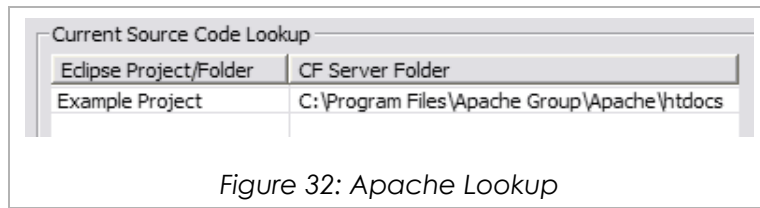


Figure 32: Apache Lookup

Both of the above examples use the default directory for each webserver but you can also configure these webserver to use any folder (Virtual Directories). In this case you can just create a lookup to that folder:

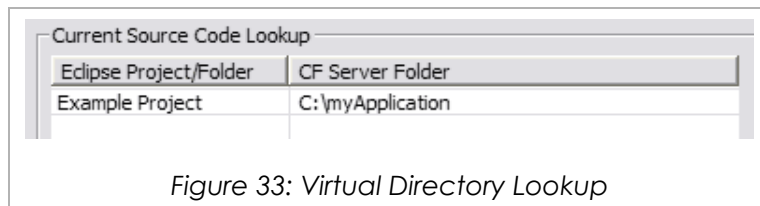


Figure 33: Virtual Directory Lookup

11.5.6 Linux and UNIX Setup

For simplicity, all of the examples so far have used a CF server on a Windows machine but FusionDebug can also debug CF servers running on Solaris and UNIX systems. Below you will see a lookup to a UNIX machine running Apache.

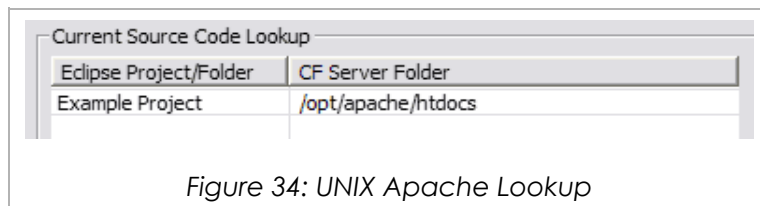


Figure 34: UNIX Apache Lookup

NOTE: You must tell FusionDebug when you are debugging a Linux or UNIX system. To do this, go to the Connect tab (next to the Source Code Lookup tab) and uncheck the item which says "CF Server is on Windows (uncheck if the CF Server is on Linux or Unix)".

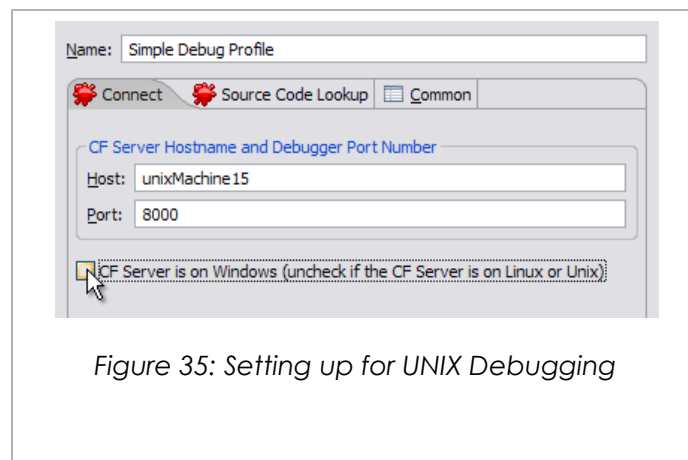


Figure 35: Setting up for UNIX Debugging

11.6 Why Are My Breakpoints Not Active?

If you look at the breakpoint view in Eclipse, you may see something like this:

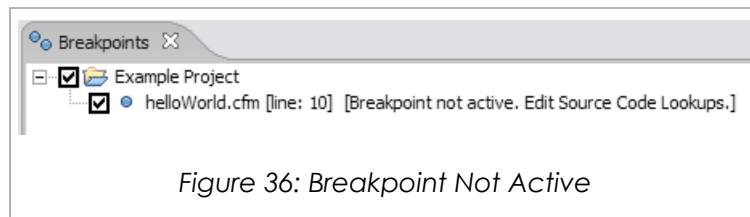


Figure 36: Breakpoint Not Active

In this case, you need to check your lookup rules in the **Source Code Lookup** tab. What this error means is that although you have told Eclipse that you want to set a breakpoint, FusionDebug couldn't work out in which file on your CF Server the breakpoint should actually be set. Go back to the Source Code Lookup tab and make sure that your project is associated with the correct folder on the server. Changes to the Source Code Lookup tab take effect immediately you select **Apply**, so this error will disappear as soon as FusionDebug can compute the real location of the breakpoint correctly.

11.7 Why Is FusionDebug Unable to Display My Source Code?

If you step into a file whilst debugging and instead see something like this instead of your source code:

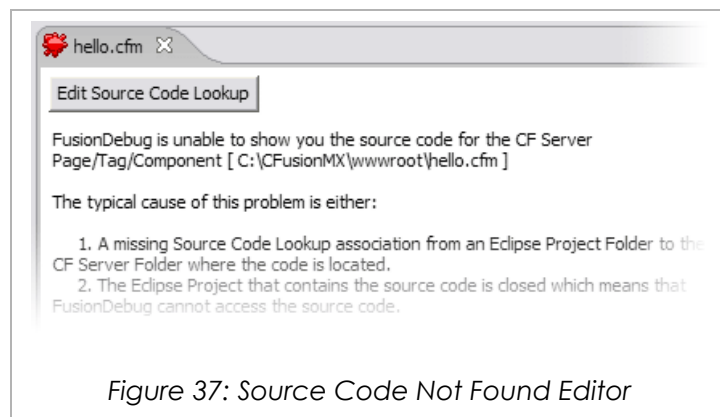


Figure 37: Source Code Not Found Editor

... then you need to check your lookups in the Source Code Lookup tab. What this page means is that the server stepped into a file but FusionDebug couldn't work out which file in your Eclipse projects was associated with the one stepped into on the server. There are two possible reasons for this:

- **The project containing the source file is closed:** If you read through the page, it will tell you which lookups were attempted and will also let you know if you have any closed projects. FusionDebug cannot show source code from closed projects. Make sure all related projects are open.
- **You are missing a mapping or have an incorrect lookup.** At the top of the page you will see a button which will take you directly to the Source Code Lookup dialog. If you click on this then you will see that the **CF Server Folder** has already been filled out. All you need to do is enter the Eclipse Project/Folder information and you can add a new lookup.

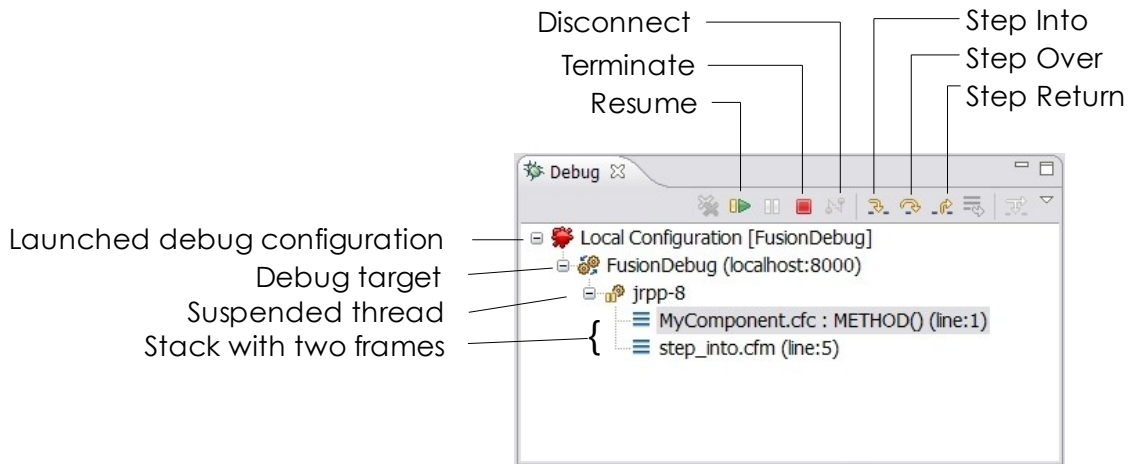


12 Reference

This section provides a feature reference for FusionDebug.

12.1 Debug View

The debug view reflects the state of the ColdFusion server being debugged.



Elements of the Debug View

Launched debug configuration	Displays the Debug Configuration used to connect to the target.
Debug target	Displays the target (machine and port) of this debugging session.
Suspended thread	If any threads are suspended because of a breakpoint hit, they are displayed here. If the thread is waiting for the server then you will see a status message after the thread name. (Listed below.)
Stack with two frames	Displays the stack of pages and CFCs involved in the current request. Each page or CFC is displayed as a frame, along with the function (if any) and line number.
Disconnect	Active on a configuration or target. Causes FusionDebug to disconnect.
Terminate	Active on a frame or thread, or configuration. When clicked: on a configuration , causes FusionDebug to disconnect. on a thread or frame , Causes the selected thread to be stopped.
Resume	Active on a target, thread or frame. When clicked: on a target , causes all suspended threads to



	<p>resume. on a thread or frame, causes that thread to resume.</p>
Step Into	<p>If the Current Instruction Pointer (CIP) is located on any tag which would cause a sub-page (CFC constructor, CFC function invoke, page function, sub-tag) to be called, Step Into instructs FusionDebug to continue stepping inside that page. If the current tag would not cause a sub-page to run, this is equivalent to Step Over.</p>
Step Over	<p>Causes ColdFusion to execute the tag currently under the CIP, then stop awaiting further stepping instructions.</p>
Step Return	<p>If the CIP is currently located inside a sub-page (e.g. A sub-tag) causes ColdFusion to finish running the code inside the sub-page and return to the caller, where execution will stop, awaiting further stepping instructions.</p>

Debug View Status Messages

The following status messages will appear after the Suspended Thread name to let you know what is currently happening:

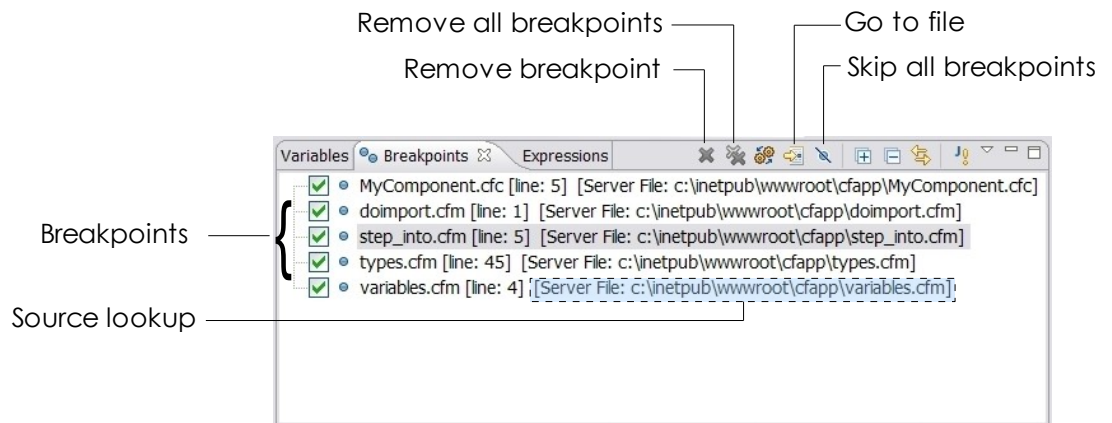
Waiting for CF Compiler	<p>ColdFusion is currently compiling a page required for the Step operation. The Debug View will update once the compiler has finished.</p>
Waiting for CF Classloader	<p>ColdFusion is searching for and loading a page, CFC or tag into memory. This often occurs after the Waiting for CF Compiler message, as part of the compile-load-run cycle.</p>
Stepping	<p>FusionDebug has asked ColdFusion to run the current line, and perform the requested Step operation, and is waiting for a report that the step has completed.</p>
Waiting for Tag Validation	<p>ColdFusion is currently validating a subtag, CFC or invoked page. The Debug View will update once the tag has been validated.</p>



12.2 Breakpoints View

This view displays all breakpoints set in the current session.

Breakpoints can be set in any supported file (ColdFusion / CFC source files) by right clicking on the required line, and selecting **Toggle Line Breakpoint**, using the key combination **ctrl-shift-B**, or with a **double-click in the ruler** to the left of the Code View.



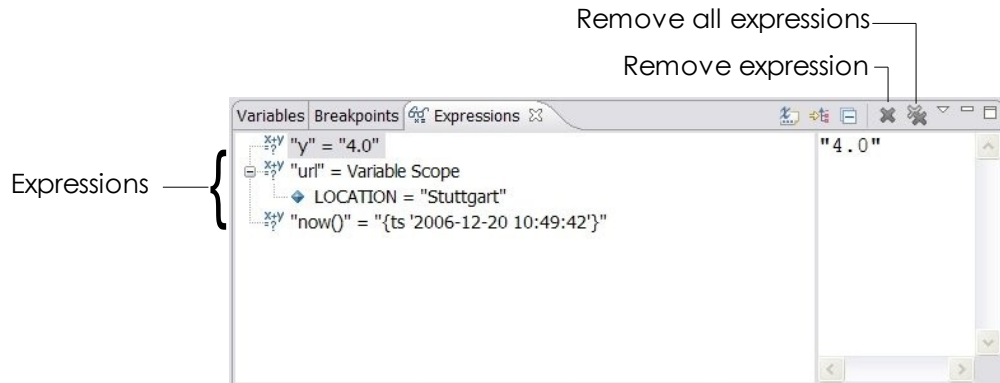
Elements of the Breakpoint View

Breakpoints	Lists all current breakpoints. Breakpoints can be individually enabled or disabled using the check mark, in which case they are displayed as an empty circle: <input type="checkbox"/> Each breakpoints specifies the file and line. Double-click the breakpoint to locate it in an editor.
Remove breakpoint	Removes the currently-selected breakpoint. Also available using the 'delete' keyboard shortcut.
Remove all breakpoints	Removes all breakpoints.
Go to File	Locates the file containing the selected breakpoint in an editor. Equivalent to double-clicking the breakpoint.
Skip all breakpoints	When selected, causes all breakpoints to be temporarily disabled. Can be used with the Resume command to run a page to completion without any intervening breakpoints firing. When activated, all breakpoints are displayed with a diagonal line:
Source lookup	Real location of the file in which this breakpoint is set. Computed using the Source Lookup Tab of the FusionDebug debug configuration. See section 11, "The Source Lookup Tab" on page 41.



12.3 Expressions View

This view displays any watched variables or expressions. Expressions can be entered by selecting them in an editor, right-clicking and selecting **Watch Expression**, or by right-clicking in the Expressions View and selecting **Add Watch Expression**.



Elements of the Expressions View

Expressions	Lists the expressions which will be evaluated.
Remove all expressions	Removes all expressions from the view.
Remove expression	Removes the selected expression from the view.

Valid Expressions

Almost all ColdFusion expressions are valid. If an expression is valid in CF Script, it is a valid Watch Expression. All variables, scope and structure references are valid:

- `url.username` – the 'username' variable from the URL scope.
- `now()` - the result of the `now()` function – the current date and time
- `y` – the value of the local variable 'y'.
- `http` – the 'http' variable scope (this will be expandable to reveal its variables)

Evaluation

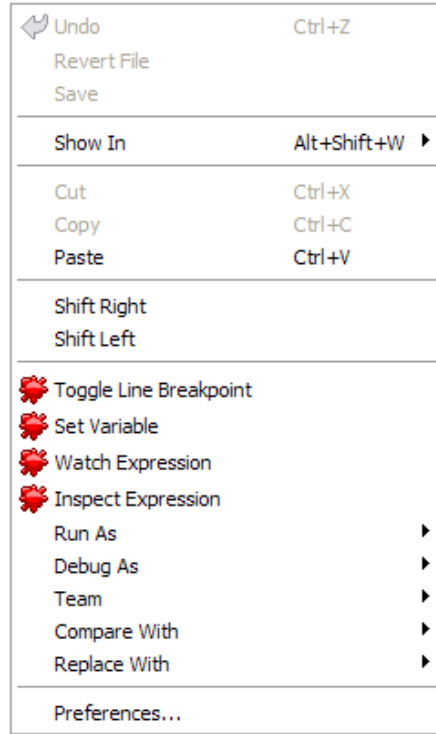
All expressions are evaluated whenever a thread pauses, either because it has hit a breakpoint or as a result of a **step** action. During evaluation, the expression will display **"(pending)"**.

Expressions can be manually re-evaluated by right-clicking the expression and selecting **Reevaluate Watch Expression**.

Expressions can be disabled from evaluation by right-clicking and selecting **Disable**. The expression will be tagged **"(disabled)"**.

12.4 Eclipse Editor Context Menu

When right-clicking within a supported source code file in an Eclipse editor, three FusionDebug elements are available, identifiable by their red FusionDebug tags.



Elements of the Context Menu

Toggle Line Breakpoint	Toggles a breakpoint at this line.
Set Variable	Allows the user to set the value of a variable.
Watch Expression	Copies the selected variable or expression to the Expressions View and evaluates it.
Inspect Expression	Immediately evaluates the value of any highlighted expression and displays it in a window. (Note: This feature is only supported on Eclipse 3.2)

Setting Variables

The **Set Variable** option allows the user to change the value of a variable.

- A variable must be selected in an editor before selecting this option.
- The value of the variable will be set in the currently-selected thread.
- The page running in that thread will immediately see the new value of that variable.
- String values must be encapsulated in double quotes ("Hello, world!")



- Expressions may also be entered – they will be evaluated and their result assigned to the variable.

This option is equivalent to running a `<CFSET var=value>` at the location of the Current Instruction Pointer.



12.5 Configuration Dialog

This section details the options available in the FusionDebug Launch Configuration dialog.

This dialog allows the user to specify one or more launch configurations: collections of parameters used to connect FusionDebug to a target ColdFusion system. It is accessible from the **Run -> Debug** menu option, or by selecting **Debug** from the debug drop-down menu in the **Debug Toolbar**, as shown in Figure 38.

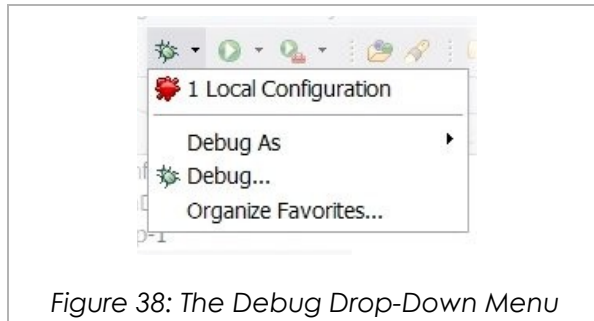
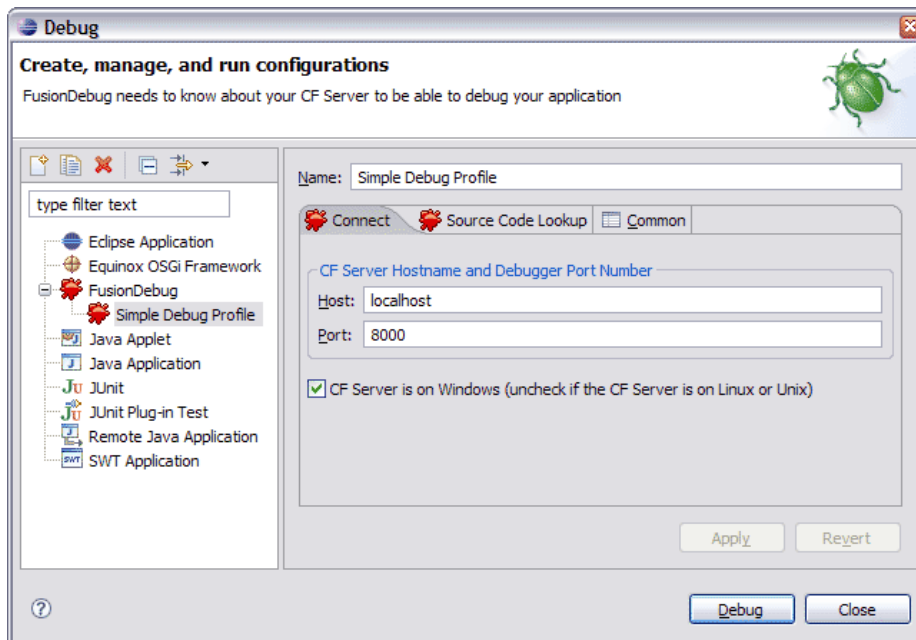


Figure 38: The Debug Drop-Down Menu

The configuration dialog for FusionDebug connections is comprised of three tabs: Connect, Source and Common. The **Name:** field is common to all three tabs, and specifies the name of this configuration. The name is used in the **Debug View** to identify which configuration is in use.

12.5.1 Connect Tab



Elements of the Connect Tab

Host

Specifies the name of the host to which FusionDebug should connect, when this configuration is launched.

Port

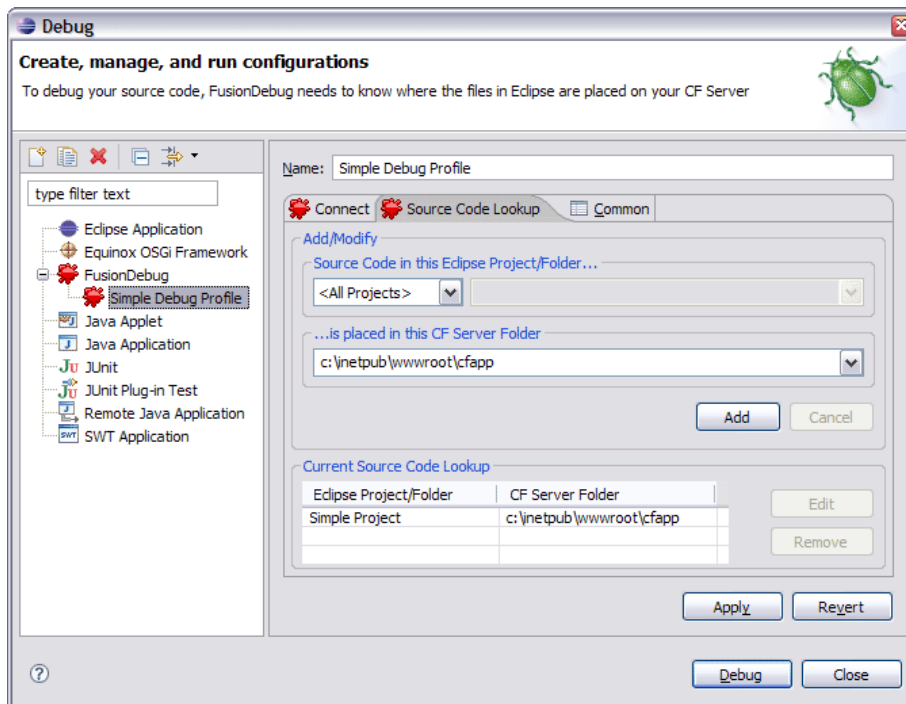
Specifies the TCP port on which the Java debug agent is listening. This is configured (for standard ColdFusion installations) in the **jvm.config** file (see section 5.4.3: "The Server Configuration Wizard" on page 26 and section 6.1: "Setting up ColdFusion for Debugging" on page 29).

CF Server is on Windows (uncheck if the CF Server is on Linux or Unix)

Specifies the operating system of the CF server. Leave checked for Windows, uncheck for Unix or Unix-like operating systems (Linux, Solaris etc.)



12.5.2 Source Code Lookup Tab



The source code lookup tab allows one to specify where FusionDebug should attempt to locate source code when a breakpoint fires, or a **step into** or **step return** action occurs. This tab is also displayed when FusionReactor can't find a source file during **Step** operations.

When connected, changes to this data take effect immediately, and the **Breakpoint View** is immediately updated with the new mappings. When disconnected, changes to this data will be reflected when FusionDebug next connects.

Elements of the Mappings Tab

Upper Panel (Add/Modify)

The upper panel allows the insertion of new lookups, and modification of existing lookups.

- The **Eclipse Project** drop-down contains all **open** projects, as well as a global **All Projects** mapping. The **All Projects** mapping is sufficient for simple projects, and is a good starting point before adding more complicated mappings.
- The **Eclipse Folder** drop-down contains all the paths for the project selected in the **Eclipse Project**. If the **All Projects** mapping is selected, this drop-down is disabled.
- The **CF Server Folder** field should contain the exact path to the mapping defined in the first two drop-downs **from the point of view of the server**. Any Windows drive letters, Unix NFS mounts or symlinks should be as viewed by the server itself. It is possible to use the drop-down arrow in this field to select previously-entered mappings.

Lower Panel (Current Source Code Lookups)

The lower panel displays existing mappings. To remove or edit an existing mapping, select it by clicking on it, and click on **Remove** or **Edit** respectively.



12.5.3 Common Tab

The common tab is not used by FusionDebug and can be ignored.

13 Troubleshooting

This section lists answers to some of the common problems encountered using FusionDebug. You can also find support information, updates and articles at our website:

<http://www.fusion-reactor.com/fusiondebug/support.html>

- **FusionDebug won't connect.** The message is: "FusionDebug could not connect to the target system (machine:port)"
 - Check you have enabled Java debugging on a port that is available, and that ColdFusion or JRun has started up correctly.
 - If you are using a non-standard port (not 8000), make sure you have correctly entered that port in the configuration dialog.
 - Make sure there are no firewalls or port restrictions in the network path between you and the target system.
- **FusionDebug won't launch.** The message is: "FusionDebug can only connect to one target at once."
 - FusionDebug 2.0 is capable of attaching to one ColdFusion instance at a time. Terminate your previous session (see "Terminate" on page 51).
- **Expressions don't evaluate.**
 - The Expressions View can handle most expressions which are valid in ColdFusion. A good rule of thumb is: if you can write it inside a CFScript block, or within "#" characters, it's a valid expression.
 - Expressions are evaluated and re-evaluated only in the context of a suspended thread. If no thread is suspended, expressions are not evaluated. If a thread is suspended, try clicking on it. This will implicitly tell the Expressions View to re-evaluate its expressions for that thread.
- **Expressions are listed as undefined.**
 - The Expressions View tried to evaluate the expression, but it either didn't return a value (for function calls) or the variable referenced in the function was not visible at the position the thread was suspended at.
- **I see lots of pages in the Debug View.**
 - These threads have suspended because of a breakpoint. Breakpoints fire whenever a request hits them. If you see many pages in the Debug View, this is because a web browser requested that page, and the page hit a breakpoint.

If you have a shared test server, we recommend disabling your breakpoints, or activating "Skip All Breakpoints" (see "Breakpoints View", on page 53) after your breakpoint has fired, so requests from someone else do not suspend.
- **Breakpoints don't fire.**
 - Check that FusionDebug is connected to your ColdFusion server (the Debug View looks similar to "Figure 2: Debug Target" on page 16).
 - Check that the breakpoint which should fire has been looked up correctly.



- The exact file in which this breakpoint is set is shown next to the breakpoint as the **Server File**. If this is not showing the correct file, or shows "**Breakpoint not active. Edit Source Code Lookups**", edit your **Source Code Lookups** using the Debug Configuration.