opentext™

OpenText RM/COBOL™

Xcentrisity®
Business Information Server
CodeWatch supplement

Version 12

Revised 2023-11-16 for version 12.20

Contents

Figures

Chapter 1. Introducing CodeWatch for Xcentrisity Business Information Server

1.1 Overview

CodeWatch™ is an integrated development environment and source code debugger for RM/COBOL applications.

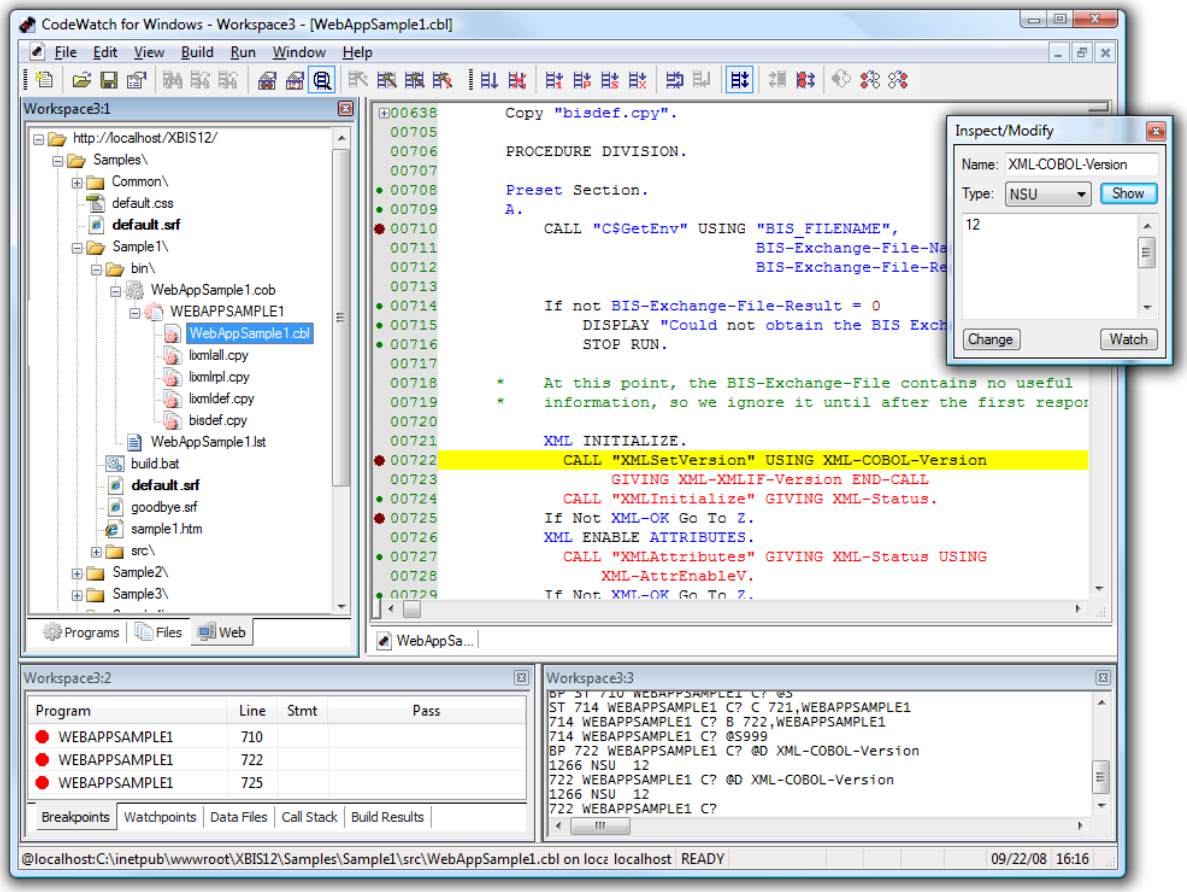


Figure 1. CodeWatch for BIS

In addition to editing, compiling, and debugging traditional RM/COBOL applications on Microsoft Windows, CodeWatch version 12 can debug service programs running under Xcentrisity™ Business Information Server (BIS) for RM/COBOL on Microsoft Windows under Internet Information Server (IIS).

This document details the BIS-specific functionality embedded within CodeWatch.

1.2 What is Business Information Server?

The Xcentrisity Business Information Server (BIS) is a web server environment that manages application sessions and makes them available via any web browser, or other web user agent that is granted access to the BIS server. BIS offers application developers an opportunity to build Service Oriented Architecture (SOA) applications incorporating legacy business data and logic freely mixed with the latest web languages and tools.

With BIS, remote users can access data, perform application functions, and execute service programs on one or multiple servers located anywhere in the world. For example, a sales force can check order status for customers during the day and enter new orders in the evening as they travel. Emergency room doctors can read patient histories on primary care physician files in another state and primary care physicians can see insurance claim's status. Bank customers can see account status, pay bills, transfer funds, and make investments, all from the comfort of their own homes. Taxpayers can have access to public records from anywhere. With BIS, any modern application architecture, function, and appearance is possible.

CodeWatch for BIS supports source debugging of such programs running on either a local or a remote IIS-based web server.

1.3 CodeWatch for BIS Features

CodeWatch for BIS can debug both browser-based and web service-based applications that are hosted by BIS on an IIS server. The IIS server may be either the current host, may be on an intranet, or may be on the internet. To ease transmissions through firewalls, all communication between CodeWatch and the BIS server are performed using the SOAP protocol over HTTP or HTTPS.

- **Browser-based applications** are launched from an HTML web page that is typically requested from a web browser. BIS can be used with any web browser. The OpenText-provided samples that are installed with BIS are examples of browser-based applications.

- **Web Service applications** are typically not launched from a browser. Instead, a program issues HTTP[S] SOAP requests to the server. No user interface is required; if there is one, it is typically provided by the client program. The BIS tutorial provided in the **samples\Tutorial1** folder contains three examples of web applications that are accessed from non-browser client programs over HTTP[S]. An example of a non-browser-based application is OpenText **BIS+WOW for RM/COBOL**. BIS+WOW service programs can be debugged by CodeWatch.

  Note that web services can be launched and consumed by a browser, but in this case, the client program is usually a JavaScript, VBScript, .NET, ActiveX® control, or Java program that is running within the browser.

1.4 Installation Requirements

CodeWatch version 12 has the same requirements as previous versions, with the following additions that are required only if Business Information Server debugging is desired:

- An installed and licensed **Xcentrisity Business Information Server for IIS** version 12 or later. BIS may be installed on the same computer as CodeWatch, or a different computer that is accessible from the computer hosting CodeWatch.

  Note that **Internet Information Services → World Wide Web Services → Security → Windows Authentication** must be installed and enabled on the IIS server using the **Turn Windows Features On and Off** application accessible from **Programs and Features**. *Basic Authentication* is insecure and is not supported.

- The CodeWatch BIS extension must be installed and enabled. The extension is installed if the **Enable CodeWatch Debugging** option is checked when a BIS web application is initially created with the **BISMKAPP** utility program. To enable debugging for an existing application directory, the application can either be re-created with **BISMKAPP**, or the CodeWatch virtual subdirectory (named **_cw** at the root of the application) can be manually created within the existing virtual directory. See the instructions on page 35 for detailed instructions on manually creating a BIS application.

1.5 Quick Start Example

The BIS installation includes a set of samples that may be used to verify that both BIS and CodeWatch are installed and operating correctly.

1.5.1 BIS Verification

To verify that BIS is installed and operational, start a web browser and enter this link:

http://localhost/xbis12/samples

(The host address **localhost**, **127.0.0.1**, or **::1** always refers to the current machine; if BIS is installed on a different machine, please use that instead.)

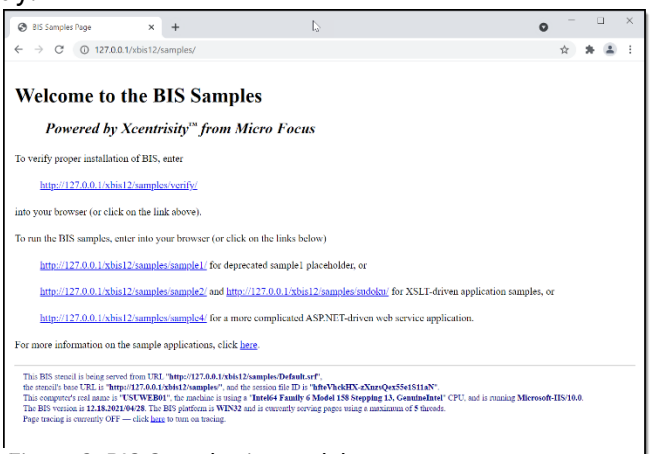


Figure 2. BIS Samples in a web browser

The page illustrated at the right should appear in the browser. If it does not, try one of these links:

> **http://*machine-name*/xbis12/samples**
> **http://localhost/xbis12/samples**

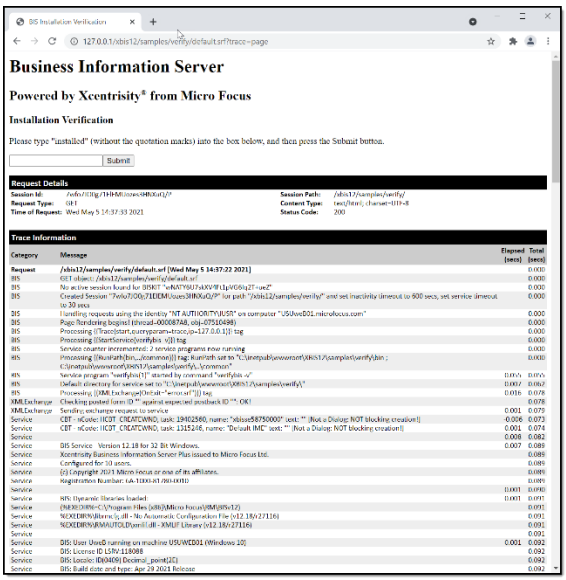
(replacing **machine-name** with the name of your host).

To see additional information, add the following to the URL: "**?trace=page**" (without the quotes). For example:

http://localhost/xbis12/samples?trace=page

If tracing is not disabled by security, you should see trace information in the browser.

Finally, run a service program: click on the first link on the page (the link that ends in **verify/**). You should see the page depicted to the right, possibly without the trace information:



Next, type the word "**installed**" in lower-case letters without quotes and click the **Submit** button. The program will confirm that you typed the word correctly, and display either a confirmation page or an error page and offer links to return to the main menu.

Either a success or failure at this point indicates that BIS is installed correctly.

1.5.2 CodeWatch for BIS Verification

Once BIS is installed and running properly, launch CodeWatch. The session wizard will appear.

1.5.2.1 Set up the Workspace

1. On the first page, select **Create New Workspace**, and under **Application Type**, choose **Xcentrisity BIS/IIS Web**. Select **Next**.

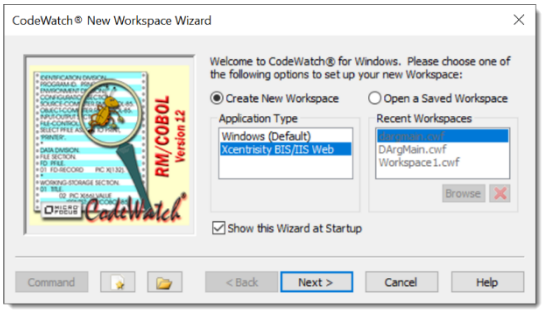


Figure 4. CodeWatch New Workspace Wizard

2. On the **Set Web Application Options** page, select **Launch BIS Application in Default Browser**, and enter this URL from above into the **URL of Start Page** edit box: http://localhost/xbis12/samples?trace=page

3. Under **Logon ID**, check **Use Current** and uncheck **Specify Debug ID** and uncheck **Debug Any Session**. Select **Next**.

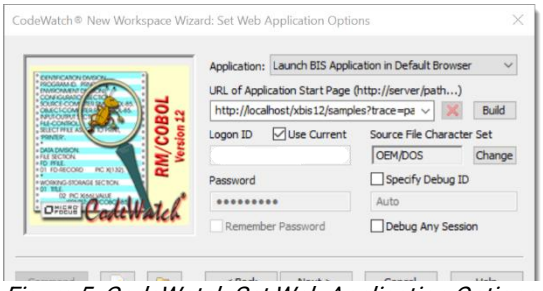


Figure 5. CodeWatch Set Web Application Options Page

On the **Test Web Connection** page, you should see a confirmation page. The key is "**All tests completed successfully.**" Click **Next**.

4. On the **Set Environment** page, optionally add the names of the directories that contain your source and COPY files (this is not required for the BIS samples). Press **Finish**.

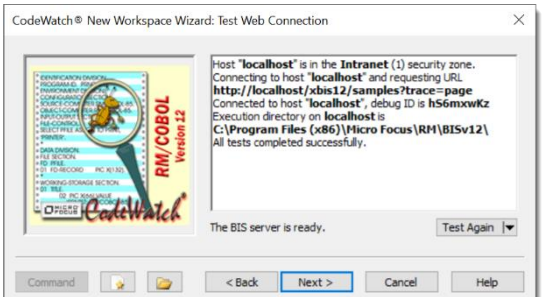


Figure 6. CodeWatch Web Application Confirmation

The main CodeWatch window will appear, and you should see a workspace window that looks like illustration on the right.

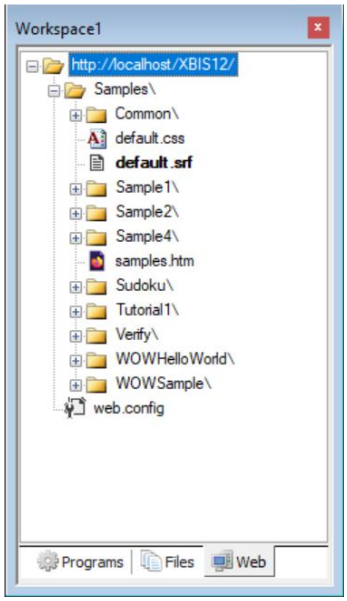


Figure 7. CodeWatch Workspace Window

Important: if the URL on the first line of the workspace window ends with **_cw**, then the CodeWatch virtual directory has been configured as a web application, rather than a virtual directory. In this case, use the IIS Administrator to remove the **_cw** virtual directory and follow the procedure described in the Appendix on page 35 to recreate the directory.

1.5.2.2 Launch the Sample Application

To start the BIS application, select **Run→Start**. Your default web browser should launch. Depending on your settings, it may prompt you for your Windows credentials. Finally, you will see the BIS Samples main page depicted in Figure 2. BIS Samples in a web browser

Click on the first link: http://localhost/xbis12/samples/verify. BIS will launch the VERIFY.COB program; CodeWatch will detect this and display the source in the editor.

CodeWatch should look like the illustration in *Figure 8. CodeWatch* , below.

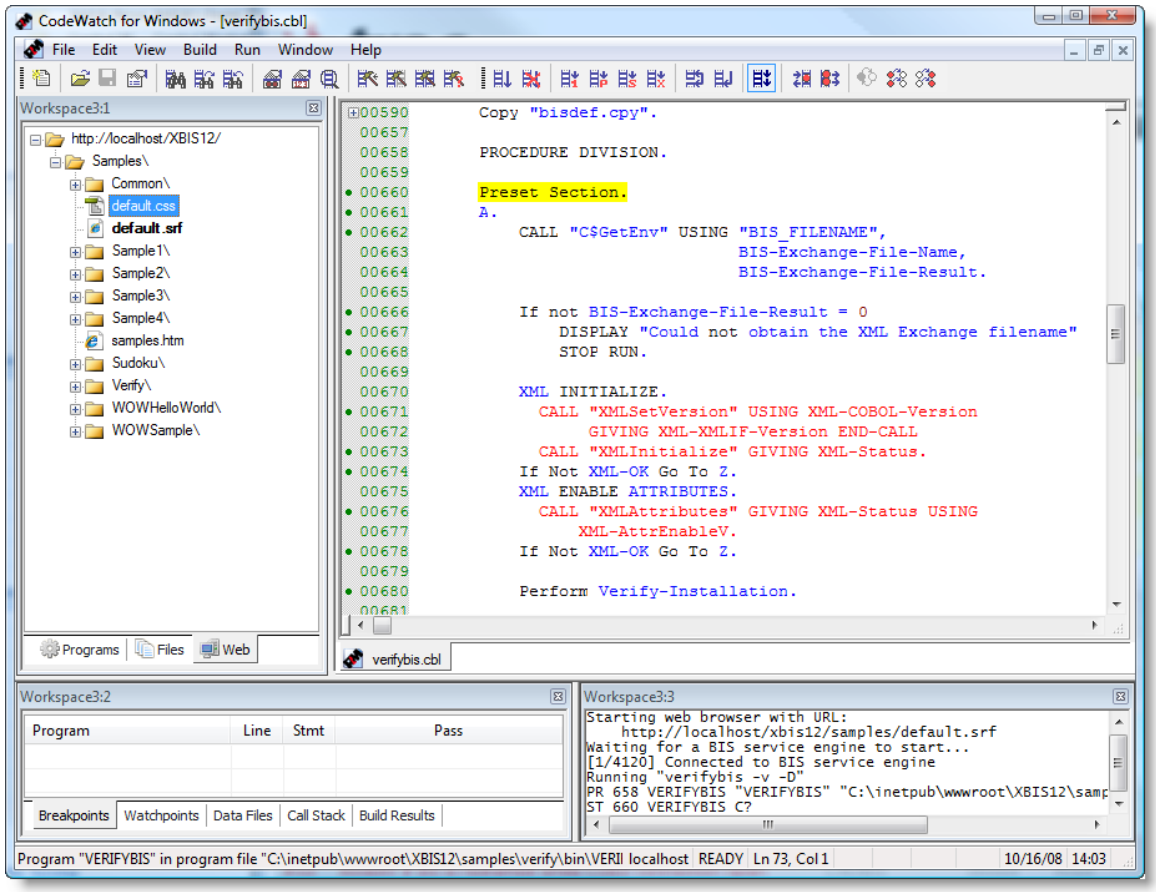


Figure 8. CodeWatch debugging a BIS application

At this point, if you press **F5** or select **Run→Start**, CodeWatch will execute until it reaches this statement:

```
Call "B$Exchange" giving BIS-Status.
```

A form will appear in the web browser. Follow the instructions in the form, and when you submit the form, the program will resume running until the **Stop Run** statement is reached.

The most important thing to note here is that the program is only loosely coupled to the web browser. The program is not notified if you close the browser, and the browser may not be notified if you terminate the program.

Also, the web browser does not automatically become the "top-most" window when interaction is requested. So when CodeWatch is paused at a call to **B$ReadRequest** or **B$Exchange** and appears to be unresponsive to additional single-step commands, be sure to check your browser window – most likely, a form has been output and BIS and the service are waiting for a response.

1.5.2.3 Saving the Workspace

You can save the CodeWatch for BIS workspace in the same way that all CodeWatch workspaces are saved. Select **File→Save** and give the **.cwf** file a descriptive name (like **BIS**

Samples). Later, you can double-click on the **BIS Samples.cwf** file in Windows Explorer to re-open the workspace, or start CodeWatch and choose "**BIS Samples**" from the recent workspaces list.

Chapter 2. Overview of CodeWatch for BIS

RM/COBOL service programs running under Xcentrisity Business Information Server are somewhat different from traditional RM/COBOL programs running under UNIX and/or Windows. The following table describes these differences.

Program Part/Area	*RM/COBOL Application on Windows or UNIX*	*RM/COBOL BIS Service Program on IIS or Apache*
Program Startup	Typically started **directly** from an icon or from a command line.	Started **indirectly** when the BIS web server receives a HTTP request for a **.srf** file that contains a **{{StartService}}** tag. The request may be issued from a web browser or a program that consumes web services.
User Interface	On Windows, interactive console output, or dialog output using COBOL WOW. On UNIX, console output only.	A web browser like Microsoft's *Internet Explorer* or Mozilla *Firefox*, or the user interface provided by the web services/SOAP client program—for example, the BIS+WOW client program.
Client Interaction	Through **ACCEPT**/**DISPLAY**, screen section, and COBOL WOW.	Interaction with BIS is through XML. The program calls **B$ReadRequest()**, which populates working storage with request variables. Once processing of the request is complete, the program calls **B$WriteResponse()** to convert working storage back into XML, which becomes part of the response to the browser or SOAP client.
Debugging	1) The runtime command line debugger 2) CodeWatch for Windows 3) Tracing through DISPLAY statements	1) DISPLAY statements that appear in the BIS trace output 2) CodeWatch for BIS version 12 or later (IIS only)
Program Termination	1) When the program exits 2) At the user's request (**Ctrl+C** or **Ctrl+Break**)	1) When the program exits 2) The service time limit is exceeded 3) The session inactivity time limit is exceeded

As detailed above, BIS service programs are started indirectly when BIS serves a stencil (a Server Response File, or **.srf** file) that contains a **{{ StartService(program-name) }}** tag. When rendered, this tag launches an RM/COBOL runtime and attaches the program to the current BIS session.

2.1 BIS Overview

A simplified schematic of BIS request processing is depicted in *Figure 9. BIS Control Flow*.

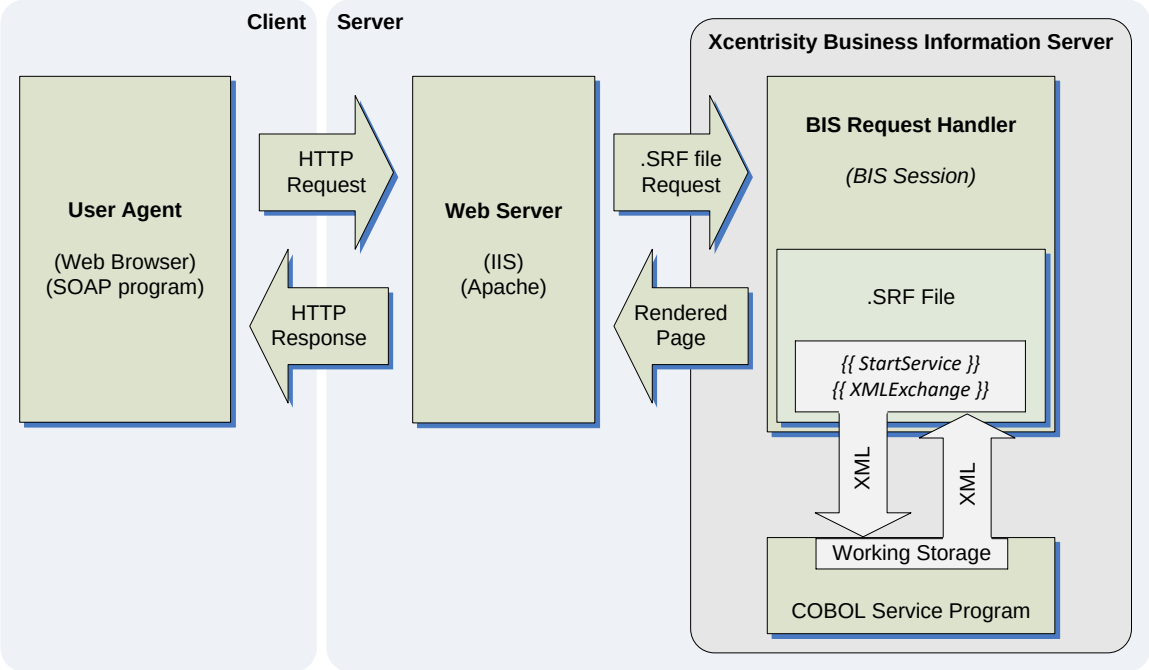


Figure 9. BIS Control Flow

The request always originates from a user agent, which may be a web browser (Internet Explorer, Firefox, Opera, Safari, etc.) or a program that consumes web services (ASP.NET web services, BIS+WOW, JavaScript within a browser, etc.). The user agent can be on the same machine (possibly even within the same web server in the ASP.NET case), or a different machine and can be either local, on an intranet, or thousands of miles away on the internet.

Here is a somewhat simplified description of how BIS handles requests:

1. The request is received by the web server, either Microsoft Internet Information Server (IIS) or Apache. (Note that CodeWatch only supports IIS.)

2. If the request is not for a BIS **.srf** file within a BIS virtual directory, the web server handles the response without invoking BIS.

   Otherwise, the web server loads the **BIS Request Handler**, which either creates a **BIS Session** or, if the request includes a valid session cookie, locates an existing BIS Session which will handle the request.

   A BIS session is automatically created by the first request from a particular user agent. The purpose of the BIS session is to preserve state across multiple BIS requests. BIS sessions have a limited lifetime: by default, expiring 10 minutes after the last request, freeing all resources consumed by that session. Session lifetime is, of course, under control of the application developer.

   After the request is linked to a session, BIS processes the **.srf** file in the context of the session, interpreting special tokens (called “tags”) that are embedded in the **.srf** file as processing instructions.

Request handler tags have this general format:

```
{{ TAG ( parameter, parameter… ) }}
```

Tags are always enclosed within a pair of curly braces (**{{…}}**). Some tags have comma-separated parameter lists; if present, these are enclosed in a single pair of parenthesis.

Tags perform various functions: **Processing Tags** set processing options, **Replacement Tags** generate inline HTML or XML, and **Action Tags** perform various actions. One action tag in particular, **{{StartService}}**, launches COBOL service programs.

3. If a **{{StartService}}** tag is encountered and the specified COBOL program is not already running in the session, then program is started in the background. Rendering then continues asynchronously to the end of the page.

4. If an **{{XMLExchange}}** tag is subsequently encountered, the request handler creates an XML representation of the request. The XML Exchange File includes form fields, query parameters, cookies, and server variables and waits for the COBOL program

5. The COBOL program calls an internal function, **B$ReadRequest**, which returns when the request is available. The program then calls **XMLImportFile**, which converts XML types to COBOL types and then populates the program's COBOL working storage with request data. Form fields, query parameters, browser cookies, and server variables are now all available to the COBOL program.

6. The COBOL program processes the request. Once processing is complete, the COBOL program calls **XMLExportFile**, which outputs working storage data items into the XML (or HTML, with a transform) response file. Finally, **B$WriteResponse** is called to send the resulting files back to the BIS request handler.

7. The request handler replaces the **{{XMLExchange}}** tag with the XML/HTML response file and sends the response to the client.

2.2 The BIS Session Monitor

When a CodeWatch for BIS workspace is created or opened, CodeWatch establishes a connection to the BIS server and creates a **BIS Session Monitor** on the server.

The BIS session monitor is a program that is loaded out of a subfolder named **_cw** on the server. This folder is created in the ***samples*** virtual directory during installation, and can be created by BISMKDIR if the **Enable CodeWatch Debugging** option is selected. Please see "Manually Configuring BIS Debugging" in the appendix starting on page 35 for more information.

2.2.1 Connecting to the BIS Server

As described above, when the CodeWatch workspace is created or opened, CodeWatch creates a BIS Session Monitor on the server using the workspace Debug ID.

Once the session monitor is running on the server, CodeWatch begins monitoring all BIS sessions that are subsequently created (existing sessions are not affected). CodeWatch will attach to a new BIS session if all of the conditions below are met:

- The BIS session is not already attached to another instance of CodeWatch.
- Debugging for the BIS session has not been disabled (see "The Debug Tag" in the *Xcentrisity Business Information Server for RM/COBOL User's Guide*).
- The BIS session has the same Debug ID as the CodeWatch monitor session or the **Debug Any Session** option has been selected in CodeWatch. See section 2.3, "The Debug ID" on page 14 for more information.

When CodeWatch attaches to a BIS session, it remains attached for the lifetime of the session.

The following illustration depicts how CodeWatch and BIS interact:

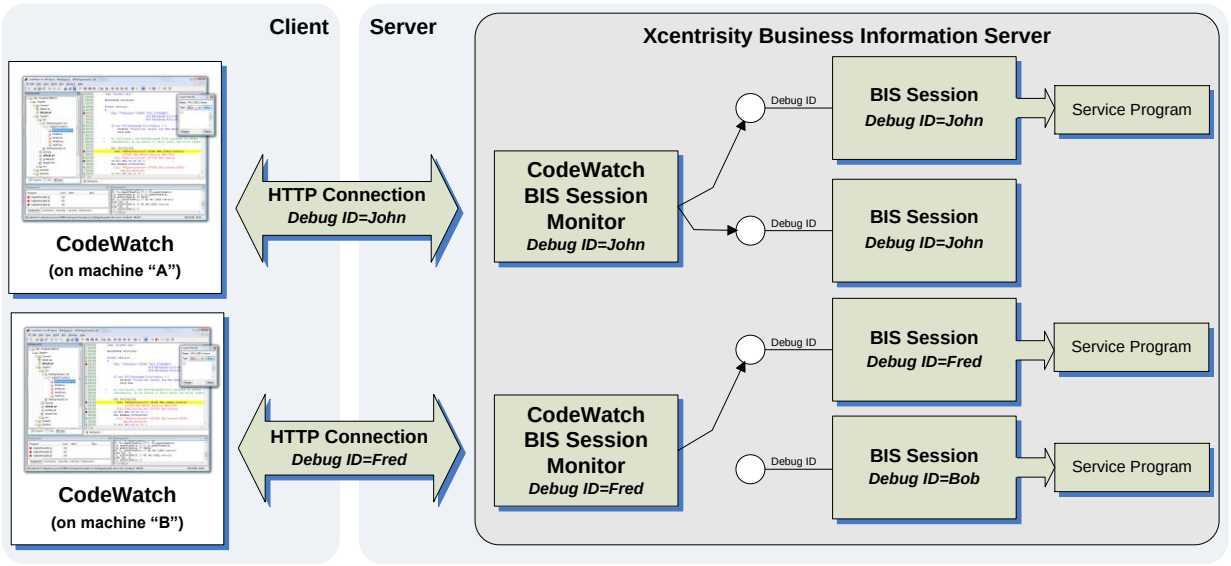


Figure 10. CodeWatch for BIS Control Flow

In the above illustration:

- There are two instances of CodeWatch (from different workstations) that are attached to the server, with Debug IDs of **John** and **Fred**.
- Each CodeWatch client is attached to a BIS Session Monitor on the server.
- Each BIS session monitor is attached to BIS sessions that are advertising the same Debug ID as the session monitor. The monitor for **John** is attached to two sessions with that ID, while the monitor for **Fred** is attached to one session. Session **Bob** runs normally, since there is no BIS Session Monitor watching for that Debug ID.

- Three of the four sessions have a running service program. Note that the monitor for **John** is attached to the two sessions that advertised that Debug ID; because CodeWatch can only debug one service at a time, the first service that starts in either of the two sessions is the one that would be debugged.

2.2.2 CodeWatch States while Connected

While CodeWatch is connected to BIS, it runs in several states that are indicated on the status bar:

- **Idle.** In this state, you can manipulate files on the server, but CodeWatch is not watching for services to begin.

- **Loading.** This state is entered when you choose **Run→Start** or press **F5**. While in this state, CodeWatch is waiting for a BIS service that is eligible to be debugged.

- **Running.** CodeWatch has attached to a program, and is debugging that program on the server.

CodeWatch can only attach to one BIS session at a time. Because BIS sessions come and go as requests are processed, the monitor session on the server must have a method to determine which BIS session to attach to. This is accomplished by a shared debug identifier, called a **Debug ID**. See section 2.3, "The Debug ID" below for more information.

2.2.3 Session Monitor Lifetime and Limits

As described above, the BIS Session Monitor is started on the server when CodeWatch connects to the server.

The BIS Session Monitor will remain on the server until one of the following occurs:

- The CodeWatch workspace is closed.
- BIS is restarted on the server—most likely due to application pool recycling.
- The web server or the machine hosting the web server is restarted.

In the latter two cases, the host will break the connection. CodeWatch will automatically attempt to reconnect, thereby creating a new BIS Session Monitor. However, because CodeWatch may be disconnected at any time, CodeWatch may be in a state where it is not possible to reconnect. In this case, please save the workspace, then close and reopen the workspace.

2.2.3.1 Monitor Limits

- On a given server, only one BIS Session Monitor with a given Debug ID may be active (including the global Debug ID). There is no limit to the number of simultaneous BIS Session Monitors that are servicing different Debug IDs.

- A given BIS Session Monitor can attach to any number of sessions at the same time, but only one service can be debugged by a given BIS Session Monitor at any one time.

2.3 The Debug ID

The CodeWatch to BIS connection is identified by an alphanumeric string called a **Debug ID**. The Debug ID is part of the CodeWatch workspace and uniquely identifies your CodeWatch session on the BIS server. This means that there can be only one CodeWatch session monitor with a specific Debug ID on the server at any particular time.

There are three ways for CodeWatch to obtain a Debug ID:

1. **Let BIS generate a Debug ID for you.** Leave the **Specify Debug ID** option unchecked in the Session Wizard or Workspace Properties. In this case, CodeWatch will ask BIS to generate a unique debug ID (one guaranteed not to currently be in use on the server). Note that the debug ID will change each time you reconnect to the server.

2. **Specify a debug ID.** In this case, be sure to choose a name that no other CodeWatch user will use on this particular BIS server. First names may not be specific enough, but something like **FirstName_LastName** or **FirstInitialLastName** may be sufficient.

   Security Note: you can improve your server's security by choosing a Debug ID that is difficult to guess. While all CodeWatch connections are authenticated, using a complex Debug ID provides an additional layer of security.

3. **Use the Global Debug ID.** This is a special Debug ID that will attach to any session. See the section entitled "**Error! Reference source not found.**" **Error! Reference source not fo und.**.

Debug IDs are case-sensitive and must range from 1 to 63 characters in length and must contain only letters, numbers, or these characters: +, =, /, \, -, _, and \.

2.3.1 The Global Debug ID

As a shortcut, CodeWatch offers an option to debug any session on the server. This option should only be selected for local test servers that are used only by a single developer at a time (note that a developer's Windows XP or Windows Vista workstation falls into this category). However, choosing this option greatly simplifies debugging web services, especially when the client is not under the developer's control.

If **Debug Any Session** is selected in the session properties, CodeWatch connects to the server with a special Debug ID that attaches to any session. Only one such BIS Monitor Session is allowed on a server at any one time, and the BIS session monitor will attach to any session that is created that does match any other BIS monitor session—that is, named matches take priority.

This feature should be used cautiously on shared servers, as it will impact other users of the server. However, debugging a web service is simpler because no preparation or session cookies are required.

2.3.2 Declaring a Debug ID in the BIS Session

The Debug ID for a BIS session is established in these ways:

- **Session Cookie**. The client sends a cookie named **DEBUG_BISKIT** with the request. If the session is not already connected to a BIS session monitor, and the value of the **DEBUG_BISKIT** matches the debug ID of a session monitor, or a global session monitor is available, then the session monitor attaches to the session.

  Note that if you are developing a browser-based application, CodeWatch launches the browser in a way that includes the Debug ID. See "**Error! Reference s ource not found.**" in **Error! Reference source not found.** on page **Error! Bookmark not defined.** for more information.

- **Debug Tag**. If the server response file (**.srf**) contains a **{{Debug}}** tag that specifies the debug ID as an active BIS Session Monitor, then the session monitor attaches to the session. Again, all subsequent sessions are ignored until the initial session ends or CodeWatch is detached from the session.

  See "*The Debug Tag*" in the *BIS User's Guide* for more information.

Chapter 3. Starting and Configuring CodeWatch for BIS

CodeWatch version 12 can be used to develop either traditional interactive RM/COBOL program, or programs that run under Internet Information Server. Use the Session Wizard to choose which type of program that you want to work on.

Traditional CodeWatch for Windows debugging is described in the CodeWatch User's Guide. This supplement to the user's guide only describes BIS-based debugging.

3.1 The Workspace Wizard

When CodeWatch for BIS is launched, you will see the Workspace Wizard. (If the Workspace Wizard does not appear, select **File→New→Workspace** or press **Ctrl+Shift+N**.)

3.1.1 Select Application Type Page

The first page that is displayed allows you to select the type of application that you will be debugging.

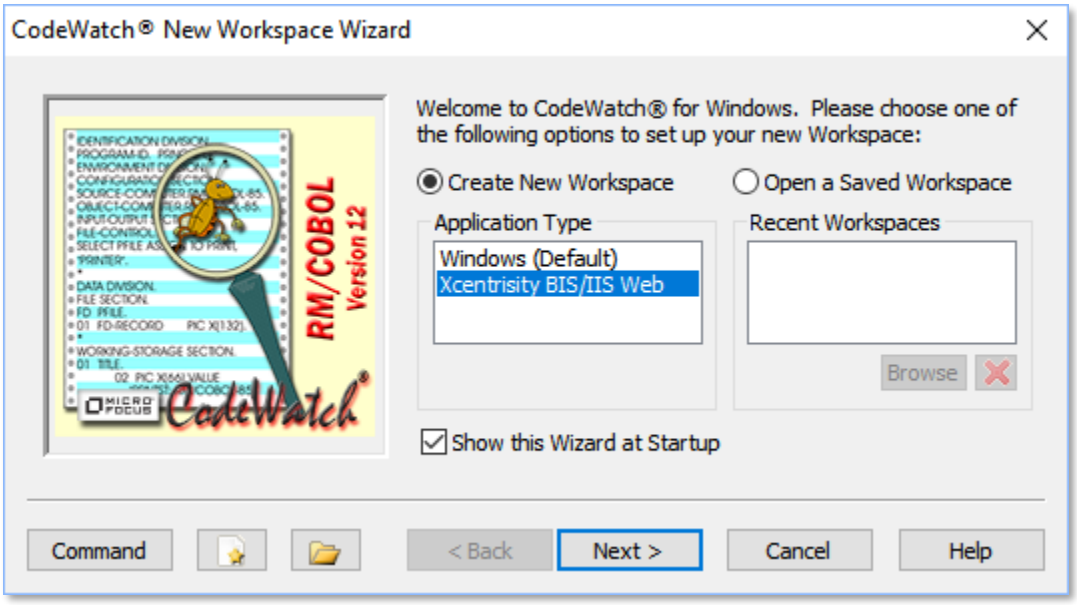


Figure 11. Choose an Application Type.

If this is the first time you are launching this program, click **Create New Workspace** and select **Xcentrisity BIS/IIS Web** from the **Application Type** list.

Otherwise, click **Open a Saved Workspace** and choose an existing workspace.

3.1.1.1 Next Button

Press the **Next** button to advance to the ***Set Web Application Options*** page.

3.1.2 Set Web Application Options Page

On this page, you specify the options that are used to launch or attach to your web application.

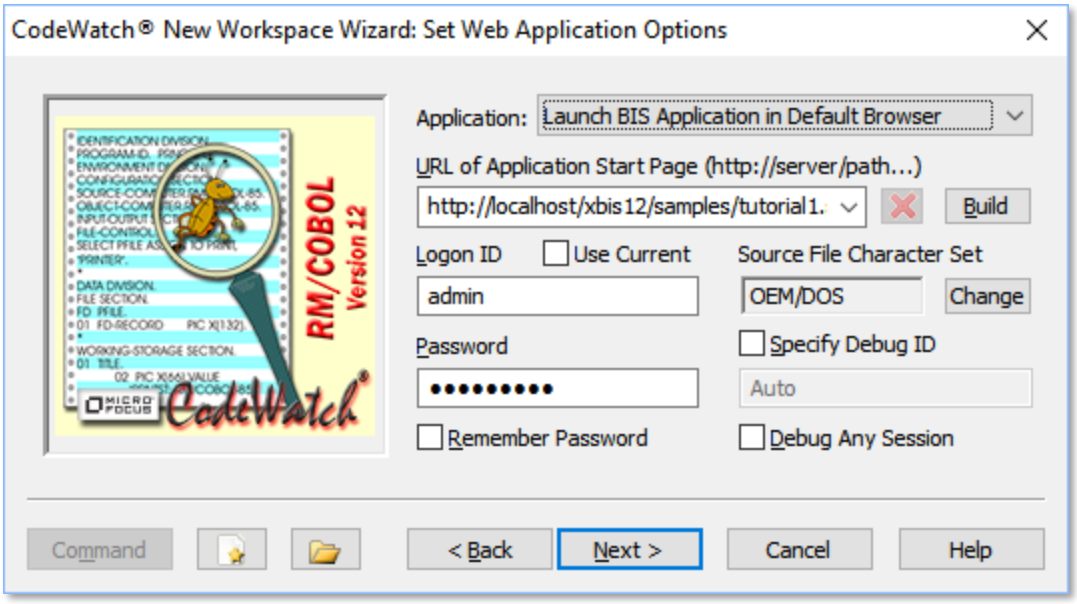


Figure 12. Web Application Options.

This page contains the following form fields:

3.1.2.1 Application

Determines how the application will be started:

- **Launch BIS Application in Default Browser**

  Launches the default Windows browser: either Internet Explorer or Mozilla Firefox.

- **Attach to BIS Application or Web Service**

  You must start the BIS application on the server. Once the BIS session is established, CodeWatch can attach to the session and will debug the web service.

This choice affects the title and usage of the next field.

3.1.2.2 URL of Appliation Start Page or URL of SOAP Service .SRF file

Enter the URL of the initial BIS page here. The usage depends on the application type: browser or SOAP client:

- **URL of Application Start Page**

  If BIS is being launched in the default browser, this combo box contains the URL that will be used to launch the application. The default browser will be launched with this URL.

- **URL of SOAP Service .SRF file (http://server/path...)**

  If BIS is providing SOAP or web services—that is, the request is being issued by a program other than a user-driven web browser—enter the URL of the BIS server response file that will serve the request.

URLs are stored in the **rmcw.ini** file, and a previously opened URL and the configuration that accompanies that URL (application type, logon ID, password, and Debug ID) may be retrieved by using the dropdown list. Press the button to remove a saved configuration.

To compose or edit a URL, either enter the URL directly or press the **Build** button to compose a URL. The following dialog will appear:

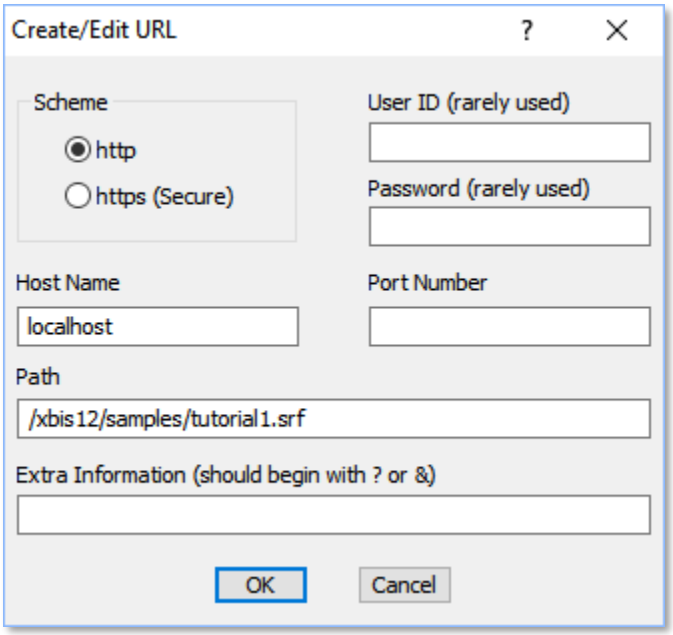


Figure 13. The Create/Edit URL Dialog Box.

See http://www.apps.ietf.org/rfc/rfc3986.html for a definition of each part of the URL.

Note: the User ID and Password fields are not the credentials that CodeWatch uses to create the BIS Session Monitor. These are application-specific credentials, and are rarely used, as they are passed in clear text as part of the URL.

Press the **OK** button to update the URL, or **Cancel** to leave the URL unchanged.

3.1.2.3 Logon ID

CodeWatch for BIS requires an authenticated connection to the server. Specifically, CodeWatch uses *Windows Integrated Authentication*, sometimes called *NTLM Authentication*. This is an optional component and must be installed on the server. Note that "Basic Authentication" sends credentials to the server "in the clear", is not secure, and is therefore not supported.

The **Logon ID** allows you to specify a user ID and password that will be used to establish an authenticated connection to the server.

✓ **Use Current**

If checked, CodeWatch will use the current logon ID and password to connect to the BIS server. The actual password is not sent—instead, a cryptographic hash of the password is sent, and is compared to a hash of the password on the server for the current logon ID. If the logon ID exists on the server and has exactly the same password, the connection is established.

If **Use Current** is not checked, you must enter a logon ID and password into the **Logon ID** and **Password** edit boxes.

Important: to use this option, both the logon ID and the password on the client machine (i.e. the machine running CodeWatch) must exactly match the logon ID on the server, and this operation must be allowed by the current security policy in effect on both machines. Also, a trust relationship must exist between the machines. If it doesn't exist, CodeWatch will offer to create this relationship on the next wizard page.

➢ **Logon ID**

If **Use Current** is not checked, enter the name of a logon ID that is valid on the server. The logon ID may be entered as a **userid** or **domain\userid.**

If **Use Current** is checked, this edit box is disabled and displays the logon ID of the currently logged-in user.

➢ **Password**

Enter the name of the password for the **Logon ID** that was entered above. This edit box is disabled if **Use Current** is checked.

✓ **Remember Password**

If an explicit **Logon ID** and **Password** are entered, check this box to store an encrypted copy of the password with the URL in the **rmcw.ini** file. If you later save the workspace, an encrypted copy of the password will also be stored in the workspace file, allowing the workspace to be opened without credentials.

The stored password will only work for the currently logged-in user. If you subsequently log in as a different user and open the workspace, you will have to supply the password.

3.1.2.4 Source File Character Set

Allows you to specify the character set that will be used to edit and compile RM/COBOL programs. The choices are **OEM/DOS** and **ANSI**. You should choose the character set that was in effect when the COBOL source file was last saved.

See "Character Set" in the CodeWatch User's Guide for more information.

3.1.2.5 Debug ID

This is where you specify the Debug ID that CodeWatch will use to identify the BIS sessions that are to be debugged.

✓ **Specify Debug ID**

When checked, enter a Debug ID into the edit box. The Debug ID must be an alphanumeric string. The Debug ID is also used as the CodeWatch connection ID, and must be unique.

Debug IDs are case-sensitive and must be from 1 to 63 characters in length and must contain only letters, numbers, or these characters: +, =, /, \, -, _, and \.

When unchecked, CodeWatch will automatically generate a debug ID that is guaranteed to be unique on the server. The generated Debug ID will be changed each time the workspace is created or opened.

✓ **Debug Any Session**

When checked, causes the CodeWatch BIS monitor to attach to the next eligible-for-debugging BIS session that is created on the server. Using this option greatly simplifies debugging, but should only be used on non-production single-user servers—for example, a developer's workstation.

3.1.2.6 Host Zone Warning

When you press the **Next** button and have specified a host that is not in your intranet or trusted sites list, you will see this message:

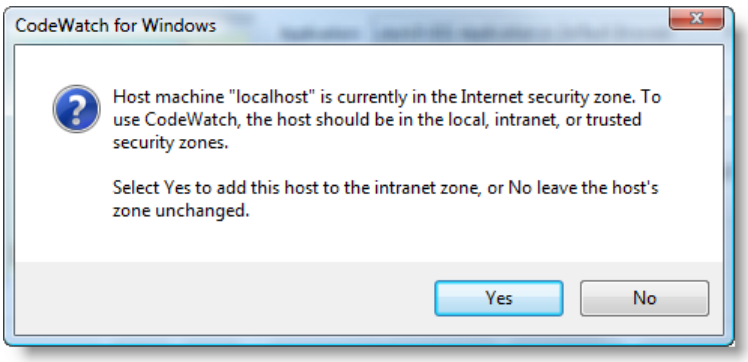


Figure 14. Host Zone Warning

If you select **Yes**, CodeWatch will attempt to add the host computer to the Intranet zone. Selecting **No** makes no changes to the security zones.

If you receive an error while attempting to add the host to the intranet zone, it is possible that an anti-virus or Internet Security program is blocking the change. In this case, you must manually add the host to the trusted zone or the intranet zone by using the **Security** tab of the **Internet Options** Control panel applet.

If the host is not in the intranet zone or trusted zone, CodeWatch may not be able to log in to the server using the logon ID and password that you entered. Press **Next** to move to the next page in the wizard and test the connection.

3.1.2.7 Next / Back Buttons

Press the **Next** button to advance to the ***Test Web Connection*** page, and the **Back** button to return to the ***Select Application Type*** page.

3.1.3 Test Web Connection Page

This page establishes and tests the connection between CodeWatch and the BIS server.

The BIS monitor connection is automatically established when you move to this page.

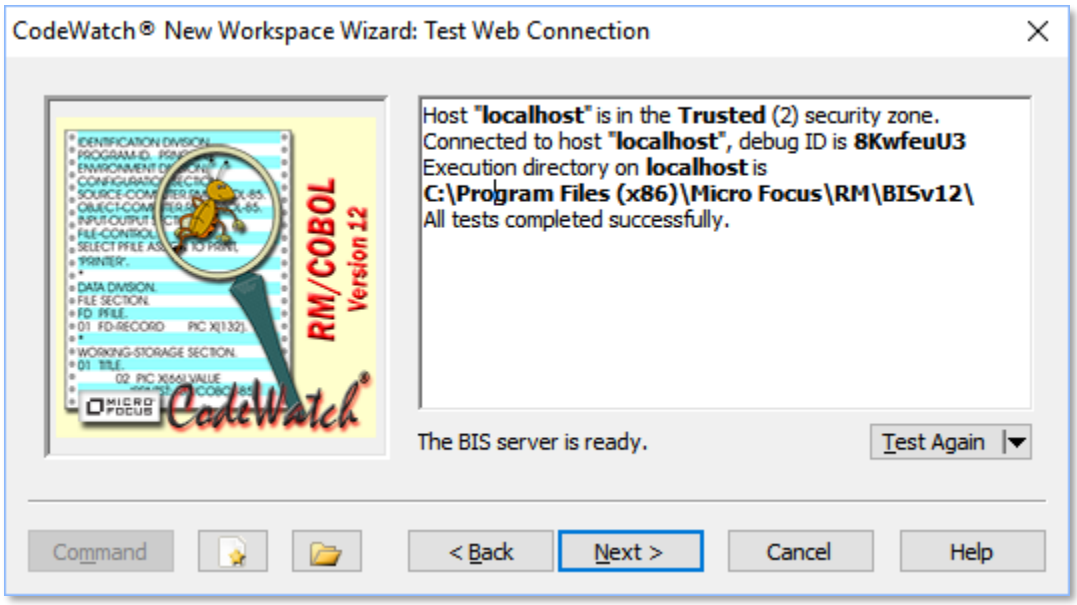


Figure 15. Test Web Connection Wizard Page

The illustration above depicts a successful test. Note the automatically generated Debug ID—in this case, "**8KwfeuU3**".

For comparison, here is a failed connection test:

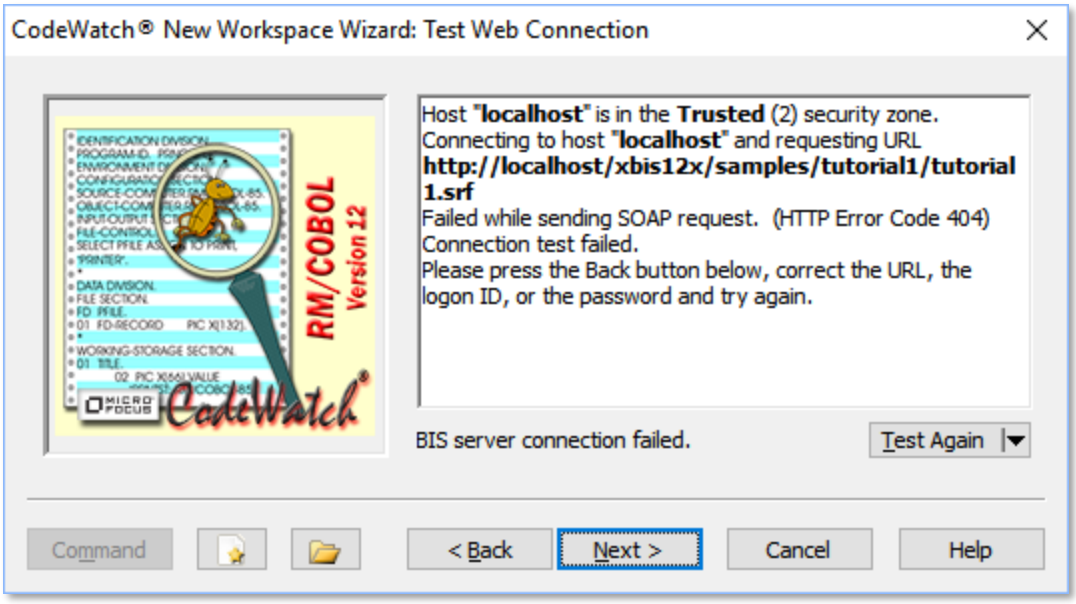


Figure 16. Failed Web Connection Example

In this case, the HTTP code **404** is the "not found" error code, which indicates that the URL is incorrect (**xbis12x** instead of **xbis12**). Note that a code of (-1) usually indicates no response from the server, which may indicates an incorrect server name or a connectivity issue.

If the connection test fails, press the **Back** button to make corrections or **Test Again** to repeat the test. Note that it may also be necessary to manually add the host to the **Intranet** or **Trusted** security zones, as described above.

3.1.3.1 Test Again Button

Press the Test Again button to connect to the BIS server, start a BIS Session Monitor and internally test the connection.

Click the arrow next to the button to display a menu:

- ✓ **Quick Test**

  Establishes the connection and exchanges a few SOAP test messages.

- ✓ **Comprehensive Test**

  Establishes the connection and exchanges a few SOAP test messages. Also performs remote I/O to test the BIS monitor.

3.1.3.2 Next / Back Buttons

Press the **Next** button to advance to the ***Set Environment*** page, and the **Back** button to return to the ***Set Web Application Options*** page.

3.1.4 Set Environment Page

This final page allows you to specify the directories that contain your COBOL source and COPY files.

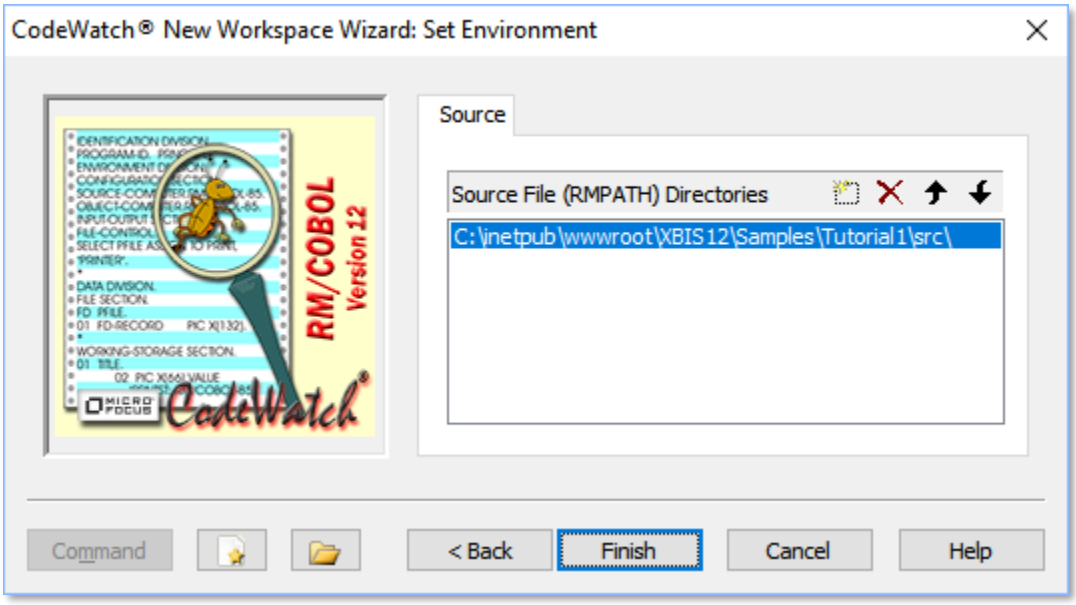


Figure 17. Set Environment Wizard Page

The list box contains directories that will be searched for source and COPY files. Directories are searched in list order. Only directories that can be reached from the Windows filesystem can currently be specified.

These buttons are used to manage the list:

- **Adds a directory to the list.**
- **Removes a directory from the list.**
- **Move selected directory up.**
- **Move selected directory down.**

To edit a pathname in the list, select the pathname and press **Space**, or double-click on the pathname with the mouse.

3.1.4.1 Finish / Back Buttons

Press the **Finish** button to complete this dialog, and the **Back** button to return to the ***Test Web Connection*** page.

3.2 Other Ways to Start CodeWatch

Besides the New Workspace Wizard, there are two other ways to start CodeWatch for BIS.

3.2.1 Loading a Saved Workspace

To load a saved workspace, select **File→Workspace→Open** and choose a CodeWatch Workspace File (**.cwf** file). If the workspace is a BIS workspace, CodeWatch will automatically

connect to the server and restore the context that was in effect when the workspace was last saved.

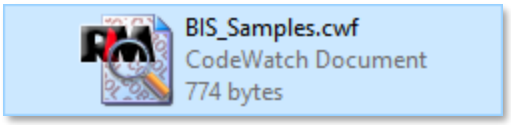
If **Specify Debug ID** was not selected, BIS will generate a new, unique Debug ID each time the workspace is opened. However, if a Debug ID was specified, that debug ID is used.

Note that, after a workspace is created or loaded, **File→Workspace→Properties** may be used to inspect or change most of the properties of a workspace.

Note that CodeWatch can also be started with the name of a CodeWatch Workspace File (**.cwf** file) on the Windows command line, accessed via the **Start → Run** command:

```
rmcw.exe C:\Users\Public\Documents\Micro
Focus\RM\CodeWatch\Projects\BIS\BIS_Samples.cwf
```

Finally, if you double-click on the icon representing a CodeWatch Workspace File (**.cwf** file) in the Windows Explorer, CodeWatch will start and load that file.



3.3 Starting CodeWatch with a URL

A URL may be specified when CodeWatch is launched from a command prompt or the Windows **Start→Run** command. For example, to run the installed BIS samples:

```
rmcw.exe  http://localhost/xbis12/samples/default.srf?trace=page
```

This always creates a browser-based project (as opposed to a web service project), and a **Debug ID** will be always be generated when CodeWatch is started in this way.

Chapter 4. Using CodeWatch for BIS

4.1 The Web Workspace Tab

For BIS workspaces, CodeWatch displays an additional tab in the workspace window: the Web tab.

This tab, shown on the right, contains a view of the BIS virtual directory on the server.

Note that the icon displayed next to each file is provided by the program that opens that type of file. Your web workspace tab may look different than the illustration.

CodeWatch does not display the Web Tab for non-BIS workspaces.

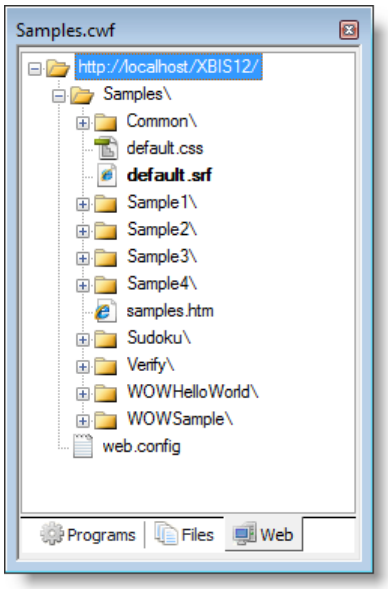


Figure 18. The Web Tab in the Workspace Window

4.1.1 Opening and Editing Files

Folders in the web tab represent directories and can be expanded and collapsed with a mouse-click. The space bar also toggles the state of folders.

Text files can be viewed and edited in the CodeWatch text editor. To open a text file, select the filename and press the **Enter** key, or double-click the filename with the mouse.

4.1.2 Examining COBOL Program Files

Program files are displayed like folders. Expanding a program file reveals the programs within that file; expanding a program reveals a list of the source and COPY files that participated in the compilation of that program.

Program files are binary files and cannot be opened by the CodeWatch editor. However, source and COPY files that were used to produce a program file can be opened, if the files are available and are reachable by CodeWatch.

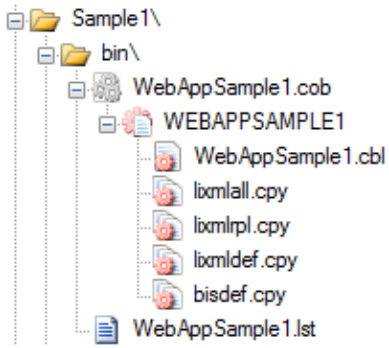


Figure 19. Expanded Program View

4.1.3 Running COBOL Programs

Because COBOL service programs are run by a web server and are launched in a variety of ways, CodeWatch cannot launch such programs directly. Instead, when you invoke **Run→Start**, CodeWatch begins monitoring the server, waiting for a program that is eligible to be debugged to start.

CodeWatch supports debugging of browser-based applications and web services-based applications. The difference:

- **Browser-based Applications** are launched from a web browser. The server response file (**.srf**) is requested directly by the browser, and the request and response type is usually a variation of HTML.

- **Web Services-based Applications** are usually launched by the program that will consume the service. The program may be running inside a browser—for example, *BIS+WOW* consists of an ActiveX control that communicates with a service program using XML—or the program may be a traditional Windows or UNIX application that internally issues SOAP requests. Web services normally use XML requests/responses.

In CodeWatch, the application type is specified in the "Set Application Options" page of the Workspace Wizard or the "Web Application" tab in the Workspace Properties window.

Browser-based and web services-based applications are started in two different ways, detailed below.

4.1.3.1 Debugging Browser-based Applications

For browser-based applications, CodeWatch automatically launches a web browser for the URL that was entered in **File→Workspace→Properties** and then waits for a COBOL service program to start on the server.

The CodeWatch log window looks like this:

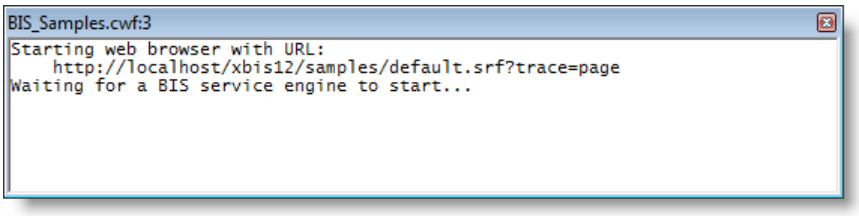


Figure 20. Waiting for BIS Application

The browser does not request the URL directly. Instead, it invokes a proxy on the server that prepares the BIS server for debugging. The BIS application program is actually launched like this:

- The URL you specified is encoded, given to the browser, which then issues a request to the BIS request monitor (the aforementioned proxy). The BIS request monitor responds with an HTTP redirect that contains this **set-cookie** header: **DEBUG_BISKIT=*DebugID***.

  This sets the **DEBUG_BISKIT** cookie in the browser, causing this cookie to be sent along with every subsequent request. This is how the server knows which session should be debugged.

- The browser executes the redirect, requesting the actual URL that you specified. It also sends the **DEBUG_BISKIT**.

- The server receives the request and, if the request is for a server response file (**.srf**), creates a BIS session. The **DEBUG_BISKIT** is then used to enable

debugging for that session and associates the session with your CodeWatch session.

If the request is not for a server response file—that is, the initial URL is for a **.htm** file, **.aspx** file, etc. then CodeWatch will continue monitoring until a **.srf** file is requested.

- Once a request finally starts a COBOL service that is eligible to be debugged (see below for the criteria), CodeWatch will attach to the service and debugging begins. Otherwise, CodeWatch will monitor the web session until an eligible service is started.

Some additional notes:

- CodeWatch will continue to wait for a service to start even across multiple BIS sessions. To terminate CodeWatch monitoring, use **Run→Stop**.

- There is no connection between CodeWatch and the browser, so closing the browser window will not terminate the CodeWatch monitoring session. This is because closing the browser window has no effect on the server.

- Session and service timeouts are disabled, by default, for BIS sessions that are being monitored by CodeWatch.

- You can use **File→Workspace→Properties** to change the URL that is initially requested. Note, however, that the new URL must refer to a page in the same IIS virtual directory that was specified when you created the workspace because the connection between CodeWatch and BIS is based on the virtual directory. If you want to work in a different virtual directory, you must close and reopen the workspace

4.1.3.2 Debugging Web Services

Because web service applications can be invoked from a wide variety of clients, CodeWatch cannot automatically launch a program to consume the web service. Instead, the BIS application session to be debugged must identify itself to the BIS Session Monitor.

There are several ways to do this:

1. If you control the web application, include a **DEBUG_BISKIT** cookie in the request that creates the session. The **DEBUG_BISKIT** must specify the **Debug ID** of the BIS request monitor.

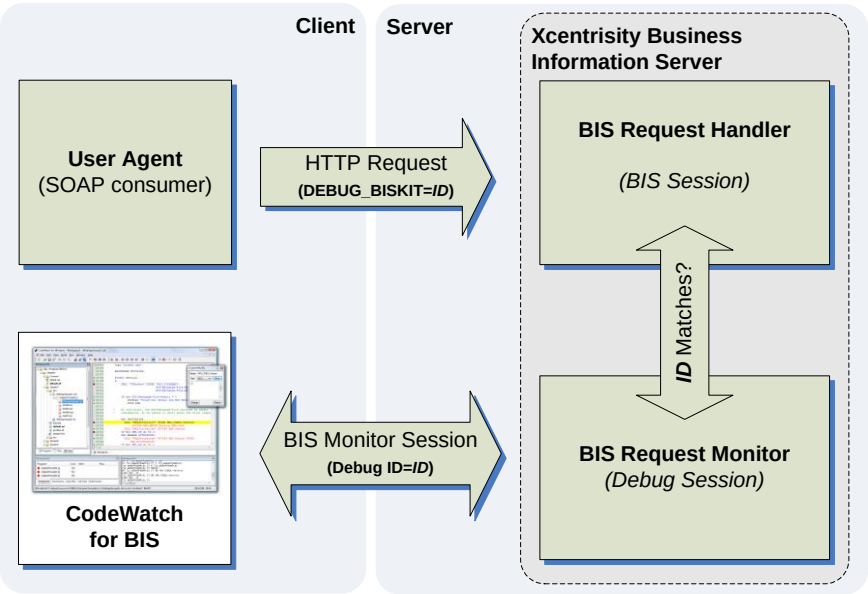


Figure 21. Web Services Debugging

This requires that you have access to the client code that sends the request, and a method to insert an arbitrary **DEBUG_BISKIT** into the request. This is the recommended technique, if you have full control over the web client application.

One example: add a form field (visible only during development) to your application that allows you to specify the value of the **DEBUG_BISKIT** cookie. If specified and a CodeWatch session with the same ID is active, the next service started in the BIS session will be debugged.

2. Add a **{{Debug}}** tag to the server response file (**.srf**) for the web service. This option requires that you explicitly specify a Debug ID in CodeWatch, and use the same Debug ID in the server response file. For example, if the CodeWatch Debug ID is "**JohnSmith**", specify this debug tag in the **.srf** file:

```
{{ Debug (ON, ID=JohnSmith, IP=ANY) }}
```

If a CodeWatch BIS Session Monitor for a debug ID of **JohnSmith** is active, and a session is created from a **.srf** with the above tag, the next service started in that session will be debugged.

3. Note that Debug IDs are case-sensitive and must range from 1 to 63 characters in length and must contain only letters, numbers, or these characters: **+**, **=**, **/**, ****, **-**, **_**, and **\.** Use the **Debug Any Session** feature. If selected, CodeWatch will debug any BIS session that is created on the server. Because any session can be debugged, this option should only be used on development servers, as any session can be debugged.

   Only one instance of CodeWatch with the **Debug Any Session** option can be attached to a given server at any one time.

Once the CodeWatch client and the server have been configured for web service debugging by using one of the techniques above, start CodeWatch and connect to the server. To begin watching for a service program, use **Run→Start**. At this point, the Web Server Options dialog that is discussed in the next section appears.

4.2 Web Server Options

The Web Service Options dialog box allows you to control which services will be debugged. It is available when debugging either browser-based applications or web service-based applications but is most useful in the latter case.

This dialog box is only available while CodeWatch is waiting for a service to begin, or is connected to a service. The dialog appears when:

- A web service is being monitored and you select **Run→Start**.
- You select the **View→Web Server Options** menu.

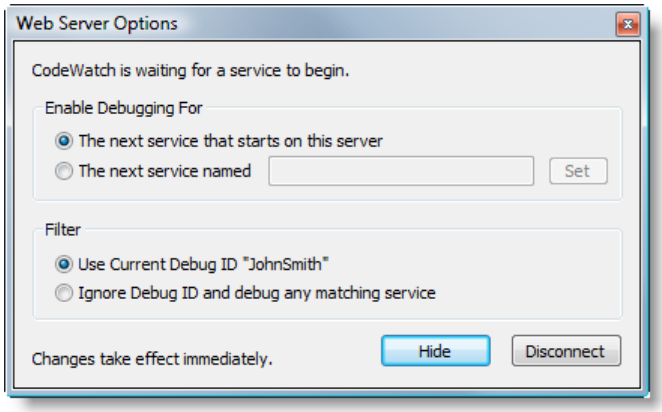


Figure 22. Web Server Options Dialog

This dialog is modeless. Press the **Hide** button to remove the dialog from view. Press the **Disconnect** or buttons to stop monitoring for a service to begin.

The following section describes the settings that this dialog controls. Note that changes take place immediately—there is no separate **Apply** or **OK** button.

4.2.1 Enable Debugging For…

This option allows you to select the service that will be debugged.

- **The next service that starts on the server**

  The next service will be debugged, regardless of the name.

- **The next service named *service-name***

  Enter the name of a service. The name must exactly match the name used in the **{{StartService}}** tag. This is useful for debugging a web service that starts different consecutive programs.

4.2.2 Filter

This option determines which services are eligible for debugging.

- **Use Current Debug ID "*name*"**

  Debugs only services running in sessions labeled with the Debug ID.

- **Ignore Debug ID and Debug any matching service**

  This is the same as temporarily electing **Debug Any Service** in the workspace properties, and is the only option available if that option was chosen. CodeWatch will monitor any session that is not attached to another instance of CodeWatch.

  This is useful for debugging web services, but note that this option should only be used on single-user servers, and never on production servers as it can impact end user services—although filtering the service name (above) can help make that a bit safer.

4.2.3 Hide / Disconnect Buttons

This dialog is open whenever CodeWatch is either waiting for a service to begin, or is attached to a service. The Hide button simply hides the dialog; the **View→Web Server Options** menu will cause the window to reappear.

Press the **Disconnect** button to stop monitoring the service, and is the same as choosing **Run→Stop** . Note that this will send a terminate signal to the service.

4.3 Server Timers

BIS uses two independent timers:

- **Session Inactivity Timeout**: the maximum period of time allowed between requests. If this time is exceeded, the session is destroyed.

  The default interval is 600 seconds (10 minutes).

- **Service Response Timeout**: the maximum period of time allowed between

  - Program startup and the first call to **B$ReadRequest** or **B$Exchange**.
  - Calls to any of these B$ functions: **B$ReadRequest**, **B$WriteResponse**, **B$Exchange**, or **B$SetServiceTimeout**.
  - Calls to any of the above **B$** functions and program termination.

  The purpose of this interval is to detect a non-responsive or runaway program and the default interval is 30 seconds.

However, these values are inconvenient for debugging, since more than 30 seconds may be required while single-stepping the program, examining variables. For this reason, CodeWatch disables these timers by default for monitored sessions.

4.3.1 Enabling/Disabling Timers

Server timers may be enabled and/or disabled with the **Run→BIS Inactivity/Response Timers** menu depicted in Figure 23 - Web Server Timers menu. The default is to disable both session inactivity and service response timers for all monitored BIS sessions. A session or service

timeout can be triggered by selecting “Simulate Inactivity Timeout” or “Simulate Service Timeout”.

Server timers may be enabled and/or disabled with the **Run→BIS Inactivity/Response Timers** menu:

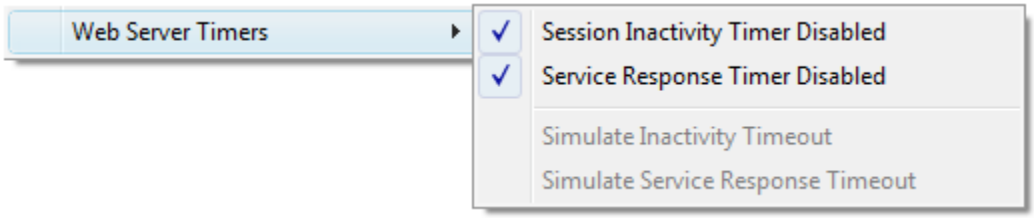


Figure 23. Web Server Timers menu

The following options are available:

- ✓ **Session Inactivity Timer Disabled**

  When checked, monitored BIS sessions will not expire (the default). When not checked, sessions expire as if CodeWatch was not attached.

- ✓ **Service Response Timer Disabled**

  When checked, BIS services will not be terminated if more than 30 seconds elapses between B$ calls (the default). When not checked, services are terminated as if CodeWatch was not attached.

- ➢ **Simulate Inactivity Timeout**

  Select this menu item to set the session timer to zero seconds.

- ➢ **Simulate Inactivity Timeout**

  Select this menu item to set the service response timer to zero seconds.

4.4 Debugging the Program

Once the **Run→Start** command has been given, CodeWatch begins waiting for a service to be launched. For browser-based applications, CodeWatch will launch the browser with the specified URL. For web service applications, CodeWatch will display the **Web Server Options** dialog.

Note that, for browser-based applications, CodeWatch does not monitor the web browser—that is, closing the browser has no effect on CodeWatch, nor on the server or service program, which continues to run.

Also note that, with the exception of compilation, all CodeWatch features that work with local files will also work with remote files.

4.4.1 Finding Source Files

CodeWatch will attempt to locate the source files for your programs on the server, using the path that was specified when the program file was created. If this path does not point at the source file, CodeWatch will then search locally for the source files.

Use the **File→Workspace→Properties→Environment** to add or remove directories from the list. The directories in the RMPATH will be searched after these explicitly-specified directories.

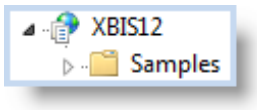
Appendix A. Manually Configuring BIS Debugging

This appendix details the steps that are required to modify an existing BIS virtual directory for debugging. These examples use the **XBIS12** directory created during installation—replace this name with the name of your virtual directory as required.

A.1 Internet Information Server (IIS) Manual Configuration

Use the steps below to enable CodeWatch debugging for a BIS virtual directory that is installed on Internet Information Server (IIS) version 7.5 or later. Note that IIS 7.0 was introduced with Windows Vista/Windows Server 2008 and is no longer supported.

1. Open the IIS administrator and expand the web tree until you reach the **XBIS12** virtual directory. Then, expand that directory. The result should look like the illustration at the right.



2. Right-click on **XBIS12** and choose **Add Virtual Directory** from the popup menu. The dialog box illustrated on the right will appear.

3. For **Alias**, enter **_cw** (this name cannot be changed). For **Physical path**, enter the location where you installed BIS – this should be one of these locations:

   C:\Program Files (x86)\Micro Focus\RM\BISv12

   or for the 64-bit version:

   C:\Program Files\Micro Focus \RM\BISv12-64

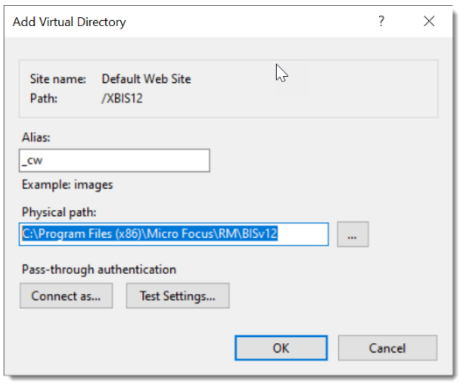
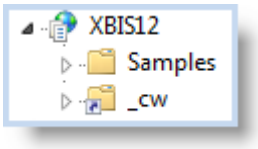


Figure 24. IIS Add Virtual Directory Dialog Box.

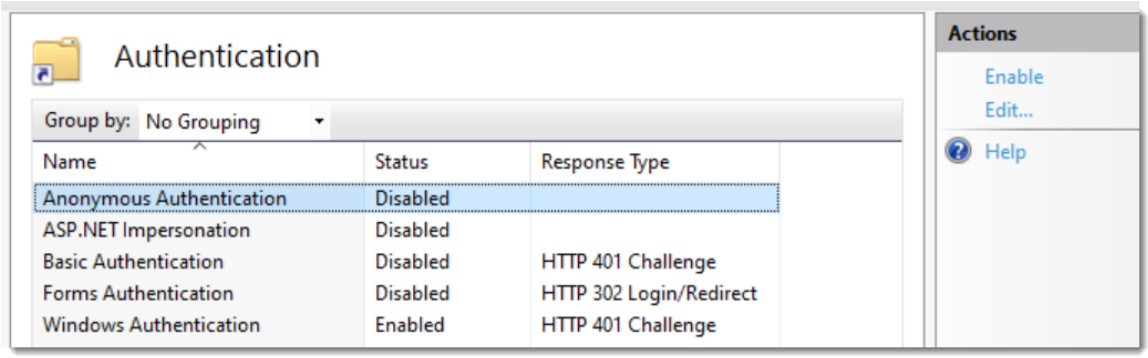
4. Click OK. A new folder will appear in the tree:



4. **Important:** To disable anonymous access to the debugger directory, follow these steps:

   a. Make sure that the **_cw** directory is selected under the **XBIS12** website.
   b. In **Features View**, double-click **Authentication**.
   c. Click **Anonymous Authentication**.
   d. Under **Actions**, click **Disable,** or right-click on **Anonymous Authentication** and select **Disable**.
   e. Select **Windows Authentication** and select **Enable**.

The display should look something like this:



WARNING: If you do not disable anonymous access to the **_cw** directory, anyone with a CodeWatch debugger will be able to connect to your server and view or modify files.

5. To verify, start CodeWatch. If the session wizard does not appear, use **File→Workspace→New**. Then, select **Xcentrisity BIS/IIS Web**, select **Launch Application in Default Browser**, and enter this URL:

```
http://localhost/xbis12/samples/default.srf
```

(or the URL of an **.srf** in your application.). Press **Next** to test the connection. If the test succeeds, debugging is properly configured.

The BIS virtual directory is now configured for CodeWatch debugging.

A.1.1 Using the APPCMD Command Line Tool to configure BIS

Both Windows Vista and Windows Server 2008 ship with a command-line tool, **APPCMD.EXE**, which can be used to configure IIS for BIS. These commands can be used instead of the above procedure to create the **_cw** virtual directory. Note that the lines in red following '>' are the expected responses.

```
C:
CHDIR C:\WINDOWS\SYSTEM32\INETSRV
APPCMD ADD VDIR  /PATH:"/_cw" /PHYSICALPATH:"c:\Program Files (x86)\Micro Focus\RM\BISv12"
                 /APP.NAME:"Default Web Site/XBIS12" /commit:APPHOST
> VDIR object "Default Web Site/XBIS12/_cw" added
APPCMD SET CONFIG "Default Web Site/XBIS12/_cw"
    /section:anonymousAuthentication /enabled:false /commit:APPHOST
> Applied configuration changes to section
> "system.webServer/security/authentication/anonymousAuthentication" for
> "MACHINE/WEBROOT/APPHOST/Default Web Site/XBIS12/_cw" at configuration
> commit path "MACHINE/WEBROOT/APPHOST"
```

The indented lines above are part of the previous command and must be on the same line. Also, note that these commands must be run as administrator—preferably from an elevated command prompt.

A complete example of how **APPCMD** can be used to set up a BIS virtual directory can be found in **BIS_MKVDIR.CMD** in the BIS installation directory.