ReadMe File for Microsoft Money 2003 Version 11.0
July, 2002

For up-to-date information on Microsoft Money, visit Money on the Web at
http://moneycentral.msn.com/money/2003/redir.aspx?mcrid=646.

How to View This Document

To view the ReadMe.txt file on screen in Notepad, maximize the
Notepad window.

To print the ReadMe.txt file, open it in Notepad or another word
processor, and then click Print on the File menu.

CONTENTS

1: Installing Money 2003 Requires Hard Disk Space On Windows Drive

When you install Money 2003, Money requires that you install
Internet Explorer 6.0, if it is not already on your computer.
This requires up to 90 MB of disk space on the drive Windows is

installed on. After installing Internet Explorer, you can choose the drive on which you want to install Money 2003.

NOTE: If you already have Internet Explorer 6.0, installing Money 2003 will require less space.

If you are low on disk space (less than 210 MB free if installing both Money and IE 6.0, less than 90 if you are just installing Money 2003) and want to free up more space on your hard drive, you can safely delete the contents of the media folder. To do this, go to the folder in which you installed Money, click the media folder and delete all the files in the folder.

If you delete these files, you will be unable to listen to the audio in the Money Setup Assistant.
It will not prevent you from using any other features of Money 2003.

If you change your mind later, you can reinstall Money, and these files will be restored

---

2: Running Money 2003 And Earlier Versions Of Money On The Same Computer

In general, Money has limited support for a side-by-side installation of Money 2003 with earlier versions of Money. Before installing Money 2003, make sure you have the CD or installation source for the previous version available. If you want to use Money 2003 while also using another version of Money, please read the following:

A. To install Money 2003 on the same computer with Money 2002

Note: If Money 2002 is installed on your computer, the default installation point is c:\Program Files\Microsoft Money.

1. Install Money 2003 on a different installation path than the earlier version of Money.
2. Make a copy of your Money 2002 data file.
3. Click the Microsoft Money 2003 Start Menu icon to start Money 2003.

Money creates a backup file named MyMoney.m10 (if your data file is named MyMoney.mny) and Money upgrades your data file. You now have two backup files you can use in Money 2002 if you choose. In addition to the copy you made in step 2, you have the MyMoney.m10 file that you can open or copy and rename.

B. To use the earlier version of Money, e.g. Money 2002

The first time you launch Money 2002 after installing Money 2003:

1. Click the Microsoft Money 2002 Start Menu icon to start Money 2002.
2. When the Password dialog box appears, click Cancel.
3. Click OK.
4. In the Open a File dialog box, select the Open/Work with an existing Money file option, and click Next.
5. Browse to the Money backup file created by Money 2003 (e.g. My Money.m10).
6. Select this file and click Open.

Once you have associated a Money data file with each version of Money, you can start the version you want by clicking the appropriate Start Menu icon.

C. To uninstall Money

1. Use Add and Remove Programs to uninstall the version of Money you want to remove.
2. If you uninstall either version, you may have to reinstall the version remaining on your computer. Make sure you have the original installation CD before uninstalling either version.

D. Other Notes:

1. Do not double-click files to open Money. Use the desktop icons or the Start menu shortcuts, so you know which version you are opening. You will need to associate the

Money files with the correct version of Money first. Otherwise, you will get the password dialog box when the older version of Money tries to open an upgraded file.

2. You cannot use different versions of Money for the same Money data file.
3. You cannot run both versions of Money at the same time.
4. Microsoft will not support more than two versions of Money installed at one time.

3: Money Not Compatible With Beta Versions of Operating Systems or Beta Versions of Internet Explorer

Please upgrade to the released version of your operating system or Internet Explorer before using Money.

4: Installing Money 2003 on machines with anti-virus software.

Some anti-virus software may interfere with Microsoft Money setup. You may want to turn off any anti-virus software before installing Money. Be sure to remember to turn it back on when Money setup is complete.

5: File Size And Money 2003

The file format used by Money 2003 lets the Money team create powerful features such as the Advisor FYI, the Lifetime Planner, and the Monthly Report (all available for Deluxe users). You might notice that your converted Money 2003 file is bigger than your old Money file.

However, you'll find that Money's compressed backup files are much smaller than your regular file. So if you need to take your data file to another computer, use the File/Backup command.

6: Internet Connections Require WinSock-Compliant ISP

Money 2003 allows you to make many connections to the Internet to gather information and advice about your finances. To connect to the Internet, you need an Internet service provider (ISP) that works directly with the 32-bit WinSock included with Windows 98, Windows 2000, Windows ME, and Windows XP.

Most full-service Internet providers (those not requiring custom software to access their service) will support the 32-bit WinSock. If you have questions about your Internet connection compatibility, ask your ISP if it supports the Windows 32-bit WinSock.

NOTE: If your ISP is America Online, make sure that you are using America Online Version 3.0a for Windows 95 or a later (not preview) version.

7: Restart Windows After Upgrading Internet Service Provider (ISP)

If you install new software from your ISP, you should restart Windows before attempting to make an Internet connection in Money. Failure to restart after upgrading your ISP software may cause miscellaneous errors in Money 2003.

8: Internet Explorer Security Settings

The following security settings are recommended when making Internet connections in Money.

1. On the Tools menu in Money, click Connection Settings.

2. Click the Security tab.

3. In the security settings area, select the Internet zone (globe icon)

4. Using the slider below, move the slider to High (most secure).

Note: Internet Explorer 6.0 or later is required to run Money, but Money still allows you to use your default browser.

9: Internet Options May Cause Money 2003 Areas To Wash Out

If you modify your Internet Explorer Multimedia settings and clear the Show Pictures check box, several places in Money may appear "washed out" and the text may be unreadable. Follow these steps to resolve the problem:

1. On the Tools menu, click Connection Settings.

2. Click the Advanced tab.

3. Scroll down until you see the Multimedia section.

4. Check the Show pictures box.

10: Money 2003 On Windows 98 Stops Responding If Modem Not Plugged In To Phone Line

Money may stop responding in the Initializing and Contacting Server portion of the Call Progress dialog box. This occurs if you are using Windows 98 and try to make a direct modem connection in Money 2003 while the modem is not connected to a phone line. You cannot use any other Windows 98 modem application.

The solution is to restart Windows and make sure that the modem is plugged in to the phone line before attempting to make a direct modem connection.

11: Windows 98 Issues With U.S. Robotics ISDN Modem And Regular Modem

If you are using Windows 98 and have a U.S. Robotics 128k ISDN adapter installed, Money 2003 may stop responding or return the following error message when you try to connect to your online banking service provider:

> Money could not complete the call due to a problem with your modem installation.

For information on how to troubleshoot this error, refer to the Microsoft Knowledge Base article, "Money: Online Banking Hangs with U.S. Robotics ISDN Adapter Installed." This article is available at http://support.microsoft.com/support/kb/articles/q160/0/51.asp.

12: Disable Remote WinSock If You Dial Up Directly To The Internet

If you are establishing Dial-Up connections directly to the Internet (using an Internet service provider or Dial-Up connection through MSN), you need to disable Remote WinSock. You may not need to disable Remote WinSock if you are dialing into a LAN or using a corporate intranet. Consult your network administrator.

1. Click Start, point to Settings, and then click Control Panel.

2. Double-click WinSock Proxy Client.
3. Clear the Enable WinSock Proxy Client check box.

13: Printer Drivers

If you experience problems printing in Money 2003, you should get an updated driver from the printer manufacturer. This is often the first step in troubleshooting printing problems.

14: Continuous Feed Wallet Checks On Windows NT

If you are using continuous feed (dot-matrix) checks on Windows NT, Money skips two checks after the fourth check in a series. For this reason, wallet-sized continuous feed checks are not recommended for Windows NT users.

If you must use continuous feed wallet checks on a Windows NT system, you should not print more than four checks at a time.

15: Printing Partial Sheets Of Checks On Sheet Feed Printers

Money defines a partial sheet of checks as when you have removed at least one of the checks from a single sheet prior to loading the sheet of checks into the printer. Money prints partial sheets of checks in landscape mode. (This orientation is similar to how you print an envelope.)

Printing partial sheets of checks (single-sheet style) is not supported by all laser or inkjet printers. Due to the feed mechanisms of some printers, it is difficult to align and print on partial sheets of checks.

The following printers have been tested and shown to have problems printing partial sheets of checks in Money:

Apple Color LW 12/660
Canon Bubble-Jet BJC 4550
Canon Bubble-Jet BJC-610
Epson Stylus 1520
Epson Stylus 800
Epson Stylus Color Esc/P 2
HP Color Laser Jet 5
HP LaserJet 5000
HP LaserJet 6P
HP LaserJet Indra
HP LaserJet 4MV
IBM 4019 Laser Printer
Lexmark ExecJet IIC 4076
Tektronix Phaser 360

If your printer has difficulty feeding partial sheets of checks in landscape mode, you can order Laser Voucher checks that never have partial sheets. You can also print Laser Standard or Laser Wallet checks in batches of three.

Depending on the feed mechanism of the printer, the following method might work to feed partial check sheets:

1. Load the partial sheet into the printer as if it were a full sheet. Push the sheet all the way into the tray.

2. On the File menu, click Print Checks.

3. Verify that the number of checks to print equals the number of checks remaining on your partial sheet. If the number of checks

to print is incorrect, follow these steps:

   a. Click Selected Checks.

   b. Select up to the number of checks remaining on the partial sheet.

   c. Click OK.

4. Make sure to enter the correct check number for the first check.

5. In answer to the question "How many checks are on the first page?", click Three. This causes Money to print in portrait orientation.

16: Tour, Video, And Audio Help Requirements

- The Money Video Help, and Audio Help are not available in all versions of Money. They are not available if you downloaded the Trial version of Money 2003 from the Web or a CD Trial version. If Money came pre-installed on your computer, these may or may not be available, depending on the version on Money installed by your computer manufacturer.

- The Videos and Audio Help are best experienced when your speakers are turned on and the volume is set appropriately.

- To view the Videos, you must have Internet Explorer 6.0 or later installed. Money installs this automatically when you install Money 2003.

- If you have manually removed Internet Explorer, you'll need to reinstall it.

- If you need to reinstall Internet Explorer, run the IE6setup.exe program from the IE folder on the Money CD-ROM.

- Your Internet Explorer security settings should be set to the default of Medium or lower to run the Tours for the first time.

17: Display Color Settings

While Money 2003 contains some color scheme options that may work in 16-color mode, Money 2003 was designed to work in 256-color mode or higher.

The Money 2003 multimedia tours and help videos are designed to work in 256-color mode or higher. If you run these in 16-color mode, the tours and videos may run slowly or may be illegible. For this reason, it is highly recommended that the tour and videos never be run using this color setting.

To check or change your color mode, follow these steps:

1. Click Start on the Windows taskbar, point to Settings, and then click Control Panel.

2. Double-click Display.

3. Click the Settings tab.

4. Check the Color Palette setting to see which mode your computer is running in. If your video card supports 256 colors, you'll see that option in the Color Palette drop-down menu.

5. Choose 256 Color from the list, and then click OK.

Windows will ask if you want to restart your computer so the change can take effect. Click Yes.

18: How To Obtain Euro Currency Symbol

If you enable Money 2003 for the euro currency, Money may not correctly display the euro currency symbol. For more information on this issue, refer to the Microsoft Knowledge Base article, "Money 2002: How to Obtain the Euro Symbol." This article is available at http://support.microsoft.com/support/kb/articles/q156/1/81.asp.

19: Quicken Versions Supported By The Quicken Conversion Wizard

Money can convert data files from Quicken versions 4, 5, 6, 7, 8, 9, 10, 11 (Quicken 2002) if you have Internet access.

20: Checks Remaining To Print Do Not Increment

If you use the Print Checks Reminder in the account register, Money does not increment the number of remaining checks the next time you print checks. There are two workarounds for this problem:

- Navigate to the Manage scheduled bills and deposits page or the account list before printing checks. If you use the Print Checks reminder in these areas, Money correctly increments the number of checks you have remaining on your sheet of checks.

- Manually update the number of checks remaining on your sheet of checks whenever you print checks.

21: Money May Appear to Stop Responding if AOL 5.0 or 6.0 Users Do Not Check the I use AOL to connect to the Internet Setting in Tools/Options/Connections.

America Online (AOL) uses unique proprietary dialer software. When using Money with AOL, you must first connect to the Internet before using Money’s online features. To connect to the Internet using AOL:

1. Start the AOL software.
2. Let the software dial and connect.
3. Minimize AOL.
   NOTE: Do not exit AOL. You must leave AOL running to provide the Internet connection.
4. Continue to use Money. The online features should now work automatically.

If you are an AOL user, and Money appears to stop responding, check to see if you are online by opening the AOL window and navigating to the Internet. If you can navigate to the Internet within AOL, but Money is still not working correctly, close Money (leave AOL running) and try again.

22: Canadian Users of Money 2003 Standard Need to Use Canadian Settings

Attention Canadian Users of Money 2003 Standard: If you create a new file in Money 2003, it’s important that you run Money with Canadian settings before creating the new file, in order to start with the appropriate set of default categories. If you are already using Money, you will know you are using U.S. Settings if you have a Taxes menu in the navigation bar. (See below for more information on Canadian Taxes.) Please refer to Help for more information on how to switch to Canadian Settings. Make sure that you click File, point to New, and then click New File to create a new file after switching to Canadian Settings.

For Canadian Tax functionality, click Categories & Payees on the Accounts & Bills menu in Money. In the left pane, click Set up tax categories to select the Tax Forms and Tax Lines that are associated with each category. When you are ready to do your taxes, click Export to Tax Software on the File menu.

23: Electronic Payments are Entered into the Account Register Using the Process Date

Money 2003 enters electronic payments into the account register using the process date. After this date, payments can no longer be canceled. However, the funds may not be withdrawn on this date based upon the payment method used by your financial institution or bill payment provider.

If you would prefer to not have Money use the processing date for entering scheduled electronic payments into the account register, clear the "Enter electronic payments in the register with the processing date" check box in Tools, Options, Online Services Tab. All future payments scheduled by your financial institution or bill payment provider will show in the register with either the due date, or the date funds are withdrawn from your account, whichever is earlier.

1. On the Tools menu, click Options.
2. Click the Online Services tab.
3. Uncheck the 'Enter electronic payments in the register with the processing date' checkbox.
4. Click the OK button.

All future payments scheduled by your financial institution or bill payment provider will show in the register on either the due date, or the date funds are withdrawn from your account, whichever is earlier.

Note: This date may be after the date when a payment may be canceled. To see the processing date, right-click the payment and select Payment Status. The processing date will be displayed. Unchecking this option does not change the date of previous transactions.

24: Money for Pocket PC Requires Upgrade to Money 2003 for Pocket PC

If you use Money for Pocket PC and synchronize it with your desktop Money file, you need to upgrade Money for Pocket PC so that it will work correctly with desktop Money 2003.

Note: you may also need to upgrade your Pocket PC operating system. Money for Pocket PC will only run on the Pocket PC 2002 operating system. To upgrade, contact the manufacturer of your device.

To install or upgrade Money for Pocket PC:

1. Go to the directory where you installed Microsoft Money 2003. (By default, the directory is C:\Program Files.)
2. Click Microsoft Money, and then click Money for Pocket PC.
3. Double-click setup.exe, and then follow the instructions on the screen.

Some downloaded versions of Microsoft Money will not include Money for Pocket PC. You can download a free copy of Money for Pocket PC on the Pocket PC Web site: http://www.microsoft.com/mobile/pocketpc/default.asp.

25: Updates to MSN Money Synchronization Require Synchronization Setup in Money 2003

If you are upgrading to Money 2003 and have used MSN Money Synchronization in previous versions, you will need to set up MSN Money Synchronization again. (MSN Money Synchronization was known as MSN MoneyCentral Synchronization in Money 2001 and earlier versions.)

The data that you synchronized in earlier versions will not automatically appear in your online portfolio or account lists until you set up synchronization in Money 2003.

Once you complete setup, your portfolio and accounts will appear online, as well as your bills.

To set up MSN Money Synchronization:

You must have a .Net Passport to use MSN Money Synchronization. If you do not have a .Net Passport, you can set one up in step 3.

1. Make sure you're connected to the Internet and signed into your Money file using .Net Passport.
2. On the Tools menu, click Access My Money Data On The Web.
3. Read the information on the Setup MSN Money Synchronization page, and then click Setup. Money will connect you to MSN Money.
4. Do one of the following:
    - If you have no accounts or investments already set up on MSN Money, go to step 6.
    - If you have only an investment portfolio set up either on MSN Money or in your Money file, you'll see the Synchronize your investment portfolio page. Select the option you want, and then click Next. Go to step 6.
    - If you already have accounts other than an investment portfolio set up on MSN Money, you'll see the Accounts already on MSN Money page with your accounts displayed in a list. Click Next and then go to step 5.
5. If you already have accounts other than an investment portfolio set up in your Money file, you should see the Identify MSN Money accounts page. Follow the instructions to match each account from MSN Money to the appropriate account in your Money file. If the account has not already been created in your Money file, create a new account. Click Next when the information on this page is correct.
6. On the Summary of your accounts on MSN Money page, review the information and then click Done.

26: MSN Money Synchronization Does Not Include Credit Card Bills Categorized as Transfers

In Money 2003, you can see your bills and deposits online if you use MSN Money Synchronization. However, it may be that you will not see your credit card bill. Many people categorize credit card bills as transfers from one account to another. MSN Money Synchronization does not include transfers so you will not see credit card bills, investment purchase bills, or any other bills or deposits that you have categorized as transfers.

27: Important Information about Online Services

In Money 2001 and previous versions, some financial institutions used an older format to exchange data. These financial institutions are listed below. (If your bank is not on the list, your services will not be affected, and you can disregard this section).

In Money 2003, all of the institutions listed below are upgrading to the newest data exchange format called OFX. This upgrade will occur when you accept an online service update from the financial institution. The upgrade is necessary if you want to use direct connection services with your bank or brokerage in Money 2003.

Note: If you do accept this upgrade in Money 2003 you will be unable to continue using direct services in Money 2001 unless you follow the steps below. It may not be possible to revert to using versions of Money prior to Money 2001.

In order to resume online banking with a Money 2001 file, you will need to copy the file "fipartnr.ini" from the PSS folder on the CD into your System folder where Money 2001 is installed, replacing the existing fipartnr.ini file. (usually at C:\Program Files\Microsoft Money\System\fipartnr.ini). Then in Money 2001, navigate to the Online Setup for your bank, hold down the Ctrl+Shift keys and click the Modify button. The button text will change to Update. Click Update and follow the conversion steps that Money will suggest.

The following banks will be upgrading to OFX for Money 2003:

American National Bank
Arizona Bank
Bank of Stockton
BankBoston PC Banking
Centura Bank
Commerce Bank, NA
Commercial Federal Bank
Compass Bank
Eastern Bank
Fleet Online Banking
Huntington Bank
JPMorgan Chase Bank
JPMorgan Chase Bank of Texas
Key Bank
LaSalle Bank NA
Manufacturers & Traders Trust
National Penn Bank
Nevada State Bank
NorwestBank
Regions Bank
Republic National Bank of NY
Smith Barney
Spectrum Connect/Reich & Tang
Standard Federal Bank
SunTrust
The Laredo National Bank
Union Bank of California
United California Bank
US Bank
Vectra Bank Colorado
Wells Fargo
Zions Bank

28: Unable to Start Money after upgrading to Windows XP

You may receive an error when you attempt to start Money after you install the Microsoft Windows XP operating system.

To resolve this issue, use the Program Compatibility Wizard in Windows XP to select a compatibility mode for Money.

1. Click Start, point to Programs or All Programs, point to Accessories, and then click Program Compatibility Wizard.
2. Click Next.
3. Click I want to choose from a list of programs, and then click Next to see a list of programs that are currently installed on your computer.
4. In the Select a program list, click Microsoft Money, and then click Next.
5. Under Select a compatibility mode for the program, click a compatibility mode, and then click Next. For example, select Microsoft Windows 98/Windows Me or Do not apply a compatibility mode.
6. In the Select display settings for the program page of the wizard, click Next.
7. In the Test your compatibility settings page, click Next to test Money with the new compatibility settings.
8. If Money starts successfully, proceed to step 9. If you receive the error message, follow these steps:
    a. Click OK in response to the error message.
    b. Click No, try different compatibility settings, and then click Next.
    c. Repeat steps 5 through 7 to select and test a different compatibility mode with which to run Money.

When Money starts successfully by using the compatibility mode that you specify, proceed to step 9.

9. Switch to the Program Compatibility Wizard, click Yes, set this program to always use these compatibility settings, and then click Next.
10. In the Program Compatibility Data page, specify whether or not you want to send program compatibility information to Microsoft, and then click Next.
11. Click Finish.

29: Installation Package Cannot Be Installed by the Windows Installer Service

When you attempt to install Money on Windows 98, you may receive an error message similar to:

This installation package cannot be installed by the Windows Installer Service. You must install a Windows service pack that contains a newer version of the Windows Installer Service.

This issue can occur if an outdated version of Windows Installer is installed.

To resolve this problem, manually start the installation of Money by following these steps:
1. Quit all programs that are running.
2. Insert your Money 2003 CD-ROM. Press and hold down the SHIFT key when you insert the CD-ROM to prevent Money Setup from starting automatically.
3. On the desktop, double-click My Computer.
4. Right-click the CD-ROM drive, and then click Open.
5. Double-click msiw9x.exe, and then follow online instructions.
6. Restart your computer.
7. Double-click My Computer.
8. Double-click your CD-ROM drive.
9. Double-click Setup.exe to install Money 2003.

30: Cannot Start Money after Setup

Money may not start properly if the installation of the System Pack was cancelled or if you chose not to install Microsoft Internet Explorer. To resolve this issue, reinstall the Microsoft Money System Pack and then Microsoft Money.

1. Insert your Money 2003 CD-ROM. Press and hold down the SHIFT key when you insert the CD-ROM to prevent Money Setup from starting automatically.
2. On the desktop, double-click My Computer.
3. Right-click the CD-ROM drive, and then click Open.
4. Double-click Syspack.msi, and then follow the instructions on the screen.

31: Money May Conflict with DirecPC Satellite Internet Connection Software.

If you use DirecPC and you receive the following error message -- Unable to initialize a required Money component -- it is because of an incompatibility between Money and DirecPC. To resolve this problem, you have to uninstall Money, uninstall the DirecPC program, reinstall Money and then reinstall the DirecPC program.

For more information about how to uninstall DirecPC software, consult the program documentation, contact DirecPC Technical Support at 1.800.347.3272 or visit the DirecPC Web site at http://www.direcpc.com.

32: Money Is Unable to Verify Your Online Sign In Name

When you attempt to sign in to Passport, you may receive an error message similar to:

Money is unable to verify your Online sign in. Money will now try to log you in using Work Offline sign in. Some online features such as background banking, will be unavailable, possibly because of network problems. To use these features, please close Money and try to sign in again later.

This issue can occur when you set the Privacy setting in Internet Explorer to High or Block All Cookies.

To resolve this issue, make sure that you are connected to the Internet. If that does not resolve the issue, follow these steps.

NOTE: Because there are several versions of Windows, the following steps may be different on your computer. If they are, please consult your product documentation to complete these steps.

Allow Cookies in Internet Explorer

1. Start Internet Explorer.
2. On the Tools menu, click Internet Options.
3. Click the Privacy tab.
4. Drag the slider to Medium.
5. Click Apply, and then click OK.

More Information

A cookie is a file that is created by an Internet site to store information on your computer, such as your preferences. For example, if you inquire about a flight schedule at an airline's Web site, the site might create a cookie that contains your itinerary or a record of the pages that you looked at within the site that you visited. The information in the cookie can help the site customize the view for you the next time that you visit.

33: Money May Conflict with Firewall Software

When you attempt to download online stock quotes in Microsoft Money, you may receive an error message similar to:

Money was unable to connect to the online quotes server. Check your Internet connection and try the operation again.

This behavior can occur if third-party firewall software such as Norton Personal Firewall or Zone Labs' ZoneAlarm is installed on your computer. To resolve this behavior, change the settings of your firewall software product to give programs such as Money permission to download without interference. Consult the firewall software help system or documentation to find out more.

For more information about Norton Personal Firewall, visit the following Symantec Web site at http://www.symantec.com/sabu/nis/npf/supportmenu.html.

For more information about ZoneAlarm, visit the following Zone Labs Web site at http://www.zonelabs.com/services/support.htm.

Authors

Au_ID	Year Born	Author
1	4/21/1963	Kusluski, Frank
10	3/13/1960	Hellstern, Albert

Xtreme Customers - Last Year's Sales

October-03

http://ic.net/~kusluski

Country	Region	City	Customer Name	Last Year's Sales
Argentina	Mendoza	Buenos Aries	Bicicletas Buenos Aires	$117,647.02
Aruba	St. George	Oranjestad	Aruba Sport	$105,205.12
Australia	New South Wales	Canberra	Canberra Bikes	$69,555.00
Australia	New South Wales	Sydney	Down Under Bikes	$89,412.00
Australia	Queensland	Brisbane	Koala Road Bikes	$53,581.22
Australia	Tasmania	Hobart	Tasmanian Devil Bikes	$78,446.33
Australia	Victoria	Churchill	Bruce's Bikes	$78,446.33
Australia	Victoria	Melbourne	Kangeroo Trikes	$119,474.00
Australia	Western Australia	Perth	Peddles of Perth	$80,311.33
Austria	Salzkammergut	Salzburg	Piccolo	$103,650.00
Bahamas	New Providence	Nassau	Beach Cycle and Sport	$38,952.00
Bangladesh	Dhaka	Dhaka	Dhaka Bike Store	$56,788.00
Barbados	Bridgetown	Bridgetown	Barbados Sports, Ltd.	$69,555.00
Belgium	Brussels	Brussels	Belgium Bike Co.	$89,990.00
Bermuda	Hamilton	Hamilton	Royal Cycle	$65,888.00
Bolivia	La Paz	La Paz	Bicicletas de Montaña La Paz	$44,671.19
Brazil	Belo Horizonte	Minas Gerais	Brasilia Outdoors	$83,098.00

Xtreme Customers - Last Year's Sales

November-03

Country	Region	City	Customer Name	Last Year's Sales
Greece	Evia Evvoia	Athens	Athens Bicycle Co.	$87,458.00
Hungary	Budapest	Budapest	East Budapest Sports	$69,555.00
India	New Delhi	New Delhi	India Cycle and Co.	$66,888.00
Indonesia	Jawa Bara West Jav	Jakarta	Jakarta Sunrise Sports	$45,871.00
Ireland	Dublin	Dublin	Greenlane Bicycles	$90,000.00
Ireland	Dublin	Dublin	Hillcrest Cycle	$51,230.00
Ireland	Dublin	Dublin	Dublin Cycle Centre	$89,756.00
Israel	Tel Aviv	Tel Aviv	Tel Aviv Outdoors	$77,889.00
Italy	Lazio	Rome	Brescia Mountainers	$59,877.00
Italy	Lombardia	Bergamo	Magazzini	$103,658.00
Italy	Lombardia	Milan	Milano Bike Store	$65,849.00
Italy	Piedmonte	Torino	Cycle City Rome	$140,560.00
Jamaica	Kingston	Kingston	Island Bikes	$88,745.00
Japan	Hiroshima Ken	Hiroshima	Nogoya Paths	$68,952.00
Japan	Hyogo Ken	Kobe	Yokohama Biking Fans	$78,027.00
Japan	Kanagawa Ken	Yokohama	Totory Ken Cyclists	$88,859.00

Xtreme Customers - Last Year's Sales

December-03

Country	Region	City	Customer Name	Last Year's Sales
USA	NJ	Hightstown	Spokes for Folks	$47,228.64
USA	NJ	Hightstown	Mountain Madmen Bicycles	$88,190.52
USA	NM	Taos	Biking Bums	$119,474.00
USA	NV	Las Vegas	7 Bikes For 7 Brothers	$78,446.33
USA	NY	Rome	Against The Wind Bikes	$80,311.33
USA	OH	Blacklick	Bike-A-Holics Anonymous	$56,328.00
USA	OH	Blacklick	Uni-Cycle	$52,428.13
USA	OH	Blacklick	Karma Bikes	$119,000.00
USA	OK	Tulsa	Hangin' Two	$53,581.22
USA	OR	Tualatin	Whistler Rentals	$68,000.00
USA	OR	Tualatin	Rad Bikes	$91,954.72
USA	PA	Conshohocken	Clean Air Transportation Co.	$89,789.25
USA	PA	Conshohocken	Insane Cycle	$88,000.00
USA	PA	Conshohocken	Rocky Roadsters	$98,681.53
USA	PA	Philadelphia	Backpedal Cycle Shop	$95,162.05
USA	PA	Philadelphia	Tek Bikes	$201,568.22
USA	RI	Kingston	Road Runners Paradise	$80,747.24
USA	RI	Kingston	Has Been Bikes (consignment)	$66,789.00
USA	RI	Kingston	Cyclopath	$88,492.56
USA	RI	Kingston	On The Edge Cyclery	$89,751.45
USA	SC	Columbia	Going in Circles	$33,999.89
USA	SD	Pierre	Off Road Bikes	$78,785.00

Debug Overview

This is preliminary documentation and subject to change

Last revised: 15 August 2001

TABLE OF CONTENTS

1. Introduction

This document describes the debugging services in the Common Language Runtime (CLR). The first part outlines the goals and requirements for the services. The second part provides an overview of the CLR debugging API. Lastly, the third part gives a picture of the architecture of the debugging services.

1.1 Design Goals

The design goals for the debugging services in the CLR are as follows:

- The debugging model should be on par with what the CLR provides. That is, the model of a program as viewed through the debugging services should be similar to that of a program as presented to the runtime for execution.
- The CLR debugging API is a low-level API. It should provide the minimal functionality necessary to 'get the job done'. The API is called by debugging packages, rather than by end-users directly. There is no need therefore to perform extensive argument checking, for example – instead, it is the caller's responsibility to perform such checks
- It should be usable by existing tools. A large investment has already been made in sophisticated tools and debugging environments and the CLR debugging services should fit in with these as easily as possible.
- Changes in the execution logic when debugging versus when not debugging should be minimized. The goal is to avoid the scenario where a user cannot reproduce some behavior when debugging is enabled, and is therefore forced to debug by more primitive means.

1.2 Further Information

Debug Reference – defines the details of the debugging API

Enabling Profiling and Debugging – describes how to control JIT-attach debugging

2. CLR Debugging Requirements

2.1 Scenarios

The CLR debugging services must operate in the following scenarios. Note that CLR does not necessarily have to enable each scenario, but it must at least inter-operate with current methods of supporting them.

Out-of-process debugging

> Out-of process debugging is when the debugger is in a process other than the process being debugged. This reduces interactions between the debugger and the process being debugged, thus allowing a more accurate picture of the process.
>
> The CLR debugging services will directly support out-of-process debugging. The services handle all communication for managed code debugging between the debugger and managed portions of the debuggee process.
>
> Even though the CLR debugging API is used out-of-process, some of the debugging logic (e.g., thread synchronization) occurs in-process with the debuggee. This is an implementation detail that should be transparent to the debugger.

In-process debugging

The CLR debugging services support limited in-process debugging. The inspection parts of the debugging API are available to be used in-process. The execution control portions of the API are not available. See the Debug Reference specification for more information on which methods are available in-process.

Remote process debugging

Remote process debugging is when the debugger user interface is on a separate machine from the process being debugged. This can be useful if the debugger interferes with the debuggee when they are running on the same machine, due to limited resources, location dependencies, or bugs which interfere with OS operation.

The CLR debugging services do not directly support remote process debugging. A debugger based on the CLR debugging API still must exist out-of-process from the debuggee, so this solution requires a proxy process on the same machine with the debuggee.

Unmanaged code debugging

Since managed code can coexist in the same process with unmanaged (native) code, there will be many situations where the user will want to debug both kinds of code simultaneously.

The CLR debugging services do not directly support debugging of unmanaged code. However they are designed to coexist with an unmanaged code debugger by sharing the Win32 debugging facilities. Also, facilities are provided to support stepping through boundaries between managed and unmanaged code.

Furthermore, the CLR debugging services provide two options for debugging a process: managed only “soft attach”, and a managed/unmanaged mode “hard attach”. When soft attached, only the managed portions of a process are debugged. In contrast, when hard attached, both the managed and unmanaged portions of a process are debugged, and all Win32 debug events are exposed through the debugging services.

Mixed language environments

In component software, different components can be built with different languages. A debugger needs to understand when the code is in different languages so it displays data with the proper format, does expression evaluation with the correct syntax, etc.

The CLR debugging services do not provide any direct support for mixed language environments, since CLR has no concept of source language. A debugger’s existing source mapping facilities should allow it to map a given function to the language in which it was implemented.

Multiple processes and distributed programs

A component program can consist of cooperating components running in different processes or even on different machines throughout a network. A debugger should be able to trace execution logic between processes and machines to provide a logical view of what is going on.

The CLR debugging services do not provide any direct support for multiple process debugging. Again, a debugger using the services is free to provide such support, and existing methods to do so should continue to work.

2.2 Environments

CLR debugging facilities are available on all processors and operating systems supported by CLR

2.3 API

The CLR debugging services are implemented primarily with unmanaged code. Therefore, the API is presented as a set of COM interfaces.

The debugging API relies on the metadata API to handle inspection of static program information such as classes and method type information. The debugging API relies on the symbol store API to support source level debugging for managed code debuggers.

2.4 Functionality Overview

2.4.1 Launching or attaching to a program

CLR allows you to attach to a running program or launch a process. The CLR debugging services support just-in-time debugging by allowing attachment to a program which throws an unhandled exception. However, a program which was not run as "debuggable" may have less debugging information available. There are various ways for a program to always run itself in a debuggable mode to alleviate this problem. For details see the specification Enabling Profiling and Debugging

2.4.2 Execution control

The CLR debugging services provide a number of ways to control execution of a program. These include breakpoints, single stepping, exception catching, function evaluation, and other events related to startup and shutdown of a program.

Execution control is only provided for managed code. Debuggers that wish to perform execution control in unmanaged code must implement it separately.

2.4.2.1 Breakpoints

Breakpoints can be created by specifying the code and MSIL or native offset of the location where the break should occur. The debugger will then be notified when the breakpoint is hit. Conditional breakpoints are not directly supported; a debugger can implement these by evaluating an expression in response to a breakpoint and deciding whether to inform the user of the stop or not.

2.4.2.2 Stepping

The CLR debugging services provide a wide variety of stepping functionality. A program can step by single instruction or by a range of instructions. It can either skip over function calls or step into them. It can also step out of a function. The debugger also will be notified if an exception occurs which interrupts the stepping operation.

Although the debugging services do not directly support stepping through unmanaged code, they will provide callbacks when a stepping operation reaches unmanaged code, to hand off control to the debugger. They also provide functionality that allows the debugger to determine when managed code is about to be entered from unmanaged code.

Stepping operations may cross thread boundaries if the CLR performs the marshalling. Stepping operations do not currently recognize other forms of cross-thread or cross-process stepping (such as COM or DCOM marshalling.)

Source level stepping is not directly provided by CLR. A debugger can provide this functionality by using range stepping in conjunction with its own source mapping information. The symbol store interfaces can be used to obtain source level information. See Debug Reference for a specification of the symbol store interfaces.

2.4.2.3 Exceptions

The CLR debugging services allow a debugger to catch both first chance and last (second) chance exceptions in managed code. The thrown object is available for inspection at each point.

Native exceptions in unmanaged code are not handled by the CLR (unless they propagate up to managed code.) However, the Win32 debugging services shared with the CLR debugging services can still be used to deal with unmanaged exceptions.

2.4.2.4 Program events

The CLR debugging services notify a debugger when many program events occur. These include process creation and exit, thread creation and exit, application domain creation and exit, assembly loading and unloading, module loading and unloading, and class loading and unloading. To insure good performance, class loading and unloading events can be disabled per module if desired. Events for class loading and unloading are disabled by default.

2.4.2.5 Thread control

The CLR debugging services provide APIs for suspending and resuming individual (managed) threads.

2.4.3 Examining Program state

The CLR debugging services provide a detailed means to inspect the parts of a process which are running managed code when the process is in a stopped state. A process can be inspected to get a list of physical threads.

A thread can be examined to find out its call stack. A thread's call stack is decomposed at two levels. First it is decomposed into *chains*. A chain is a contiguous logical call stack segment containing entirely managed or unmanaged stack frames. In addition, all managed call frames in a single chain share the same CLR context. If a cross-thread call has occurred within the thread, the chains provide information to allow a debugger to easily track logical call stacks across threads.

A chain can be either managed or unmanaged. Each managed chain can be further decomposed into single stack frames. Each stack frame represents one method invocation. Stack frames can be queried to obtain the code they are executing. A stack frame can also be queried to obtain arguments, the local variables, as well as the native registers.

An unmanaged chain is not composed of stack frames. Rather, an unmanaged chain simply provides the range of stack addresses that are due to unmanaged code. It is up to an unmanaged debugger to decode the unmanaged portion of the stack and provide a stack trace.

Note that CLR debugging services do not understand the concept of local variables as they exist in source code. It is up to the debugger to map between local variables and where they are allocated (either at a variable index for MSIL functions or in a register/stack location for managed native code.)

Access to global, class static, and thread local variables is also provided.

2.4.4 Modifying Program State

Although it is an inherently dangerous operation, the CLR debugging services allows a debugger to change the physical location of the instruction pointer during execution. The instruction pointer may be changed successfully under the following conditions:

- The target instruction pointer is at a sequence point. Sequence points represent statement boundaries.
- The current instruction pointer is at a sequence point.
- The target instruction pointer is not located in an exception filter.
- The target instruction pointer is not located in a catch block.
- The target instruction pointer is not located in a finally block.
- From within a catch block, the target instruction pointer is not located outside the catch block.
- The target instruction pointer must be within the same frame as the current instruction pointer.

Variables at the current instruction pointer location will be mapped to the variables at the target instruction pointer location. GC references at the target instruction pointer location will be properly initialized.

After the instruction pointer is changed, the services mark any cached stack information as invalid and refresh the information the next time it is needed. Debuggers that cache pointers to stack information (frames, chains, etc.) should refresh this information after changing the instruction pointer.

The debugger is also allowed to modify the data of a program when it is stopped. Local variables and arguments in a function that is currently running can be changed in a manner similar to how they are inspected. Also, fields of arrays and objects can be updated, as well as static fields and global variables.

2.4.5 Edit and Continue

Edit and Continue is a feature which makes it possible to be in the middle of a debugging session, edit source code, recompile the modified source, and continue the debugging session without having to rerun the executable from the start. From a functional perspective, Edit and Continue implies the ability to modify the code that is running in the debugger while preserving the rest of the run time state of the executable being debugged.

Note: Edit and Continue mode is not supported in first release of the product. However, the information in this section is retained, for information.

It is intended to support the following scenarios in a future release:

Compiler Support. This scenario allows a debugger process to request a snapshot of the metadata for the running process, and have this marshaled back to the compiler to use. The compiler will be able to use the metadata APIs to open the metadata in Edit and Continue mode. This means that no tokens will move. Hence, if the code generation is deterministic, no changes should be perceived except those made by the user. Adding new metadata and having it reflected in the running process is not part of this scenario.

Replace an Active Method. The programmer makes a change to a JIT-compiled, active method and the change is reflected. The debugger will support hijacking of the active method for the EE to know when it hits the top of the stack. The EE will perform the required stub work to swap out the method. The common allocator will throw away the old method and the JIT compiler will compile the new method.

Adding Static Data. The programmer adds a new text string for rdata inclusion and then updates the program to use ldstr. The same holds for other instructions. This requires the EE to return a valid RVA base for the snapshot call, recognize the new data, and put the new data into the scratch data area of the running process.

Adding Local Variables. New variables are added to an existing method. This requires a new standalone signature in the metadata describing the new variable. The EE and JIT compiler extend the stack frame for a method which gets a new variable (layout of the original stack must remain fixed – this is by design).

Add a Static Method to a Class, Call It. Add a static method to an existing class definition. This requires the metadata to be extended with the new method definition and the method layout to be changed to include knowledge of the new method.

Add non-Virtual Method to a Class, Call It. This is same as the previous scenario, but only allow for non-virtual, non-static method to be added to the class. This change requires the method table layout to have knowledge about the new method, and allow calls to the new method.

Add Virtual Method to a Class, Call It. This is same as the previous scenario, only allow for a virtual method to be added to the class. This change requires the method table layout to have knowledge about the new method, and allow calls to the new method. Space must be reserved in the v-table layout to allow for the new method to be added.

Add Virtual Method to Parent Class, Call It. This is same as the previous scenario, only the new method is in a parent class for which an instantiated instance of a derived class exists. Extra space must be reserved in the parent class's v-table to allow up to some maximum number of new methods to be added this way. This change requires new metadata and changes to the EE for method table layout.

Add Pinvoke Method to a Class, Call It. This is same as the previous scenario,, only for adding an Pinvoke surrogate value with full metadata. This change requires new metadata and some support in the EE to fault in the new library (if not already there) and the new entry point.

Override Method to Hide. The programmer may add a method override in a derived class that hides the parent method. The EE doesn't track all existing call sites that are made out of date. In theory the source dependency rules would force those targets to be recompiled and hence compiled with the JIT compiler using the proper reference.

Remapping Breakpoints. Allow breakpoints to be set in a method, then force the method body to be changed while keeping the breakpoints. This requires the breakpoints to be removed from the old method and mapped into the new method. As for any case where the method body changes, the debugger also needs to track the new JIT sequence point map. The compiler provides a map from the old MSIL sequence points to the new MSIL sequence points.

Add a New Class. The programmer may add a new class definition to a module. This change requires new metadata and EE to add the class to the delay load list.

Add a New Field to a Class. The programmer may add a new field to an existing class definition, of which there may be active instances. The change involves extending the metadata to include the new field, having the EE include this extra data, and any required work to provide the JIT compiler access to the new field. There may also be GC work to track what could be a new reference field.

The CLR does not provide functionality to delete methods and fields from a class. If a debugger supports deletion of methods and variables, it is up to the debugger to ensure that the deleted methods and fields are not used. A debugger could "tombstone" a method by using Edit and Continue to replace the deleted method with a method that throws an exception when it is invoked.

Edit and Continue operations may only be performed when the CLR is in a sane state with all managed threads stopped at safe points.

JIT Restrictions

In general, under Edit and Continue, the JIT-compiler tries to produce code (and stack frame) which is independent of previous versions of the method, but is still upgradeable from previous frames.

However, the frame layout depends on certain other factors and we don't put Edit and Continue restrictions on these. The JIT compiler has to handle these as gracefully as possible. All of these apply only to updating of a method currently on the stack.

- At the update point, the set of the older variables in scope has to be the same before and after. Newly added variables are acceptable.
- You can't be inside of a handler at the update point before or after.
- No Edit and Continue can be made if the older method used localloc.

2.4.6 Function evaluation

In order to evaluate user expressions and dynamic properties of objects, a debugger needs a way to run the code of the process being debugged. The CLR debugging services accomplish this by allowing the debugger to make a function or method call and have it run inside the debuggee's process.

Because this may be a dangerous operation (e.g., it may trigger a deadlock with existing code), the CLR allows such an operation to be aborted by the debugger. If the evaluation is aborted successfully, the thread will be treated as if the evaluation never happened, minus any side effects on locals, etc, from the partial evaluation. If the function calls into unmanaged code, or blocks in some fashion, it may be impossible abort the evaluation.

The CLR notifies the debugger via a callback when the function evaluation completes normally or if the function throws an exception. The result of an evaluation is available for inspection via the ICorDebugValueXXX API methods in the debugging API.

The debugging services will set up a new chain on the thread to start a function evaluation and call the requested function. Once started, all aspects of the debugging API are available: execution control, inspection, function evaluation, etc. Nested evaluations are supported and breakpoints are hit as usual.

2.4.7 Dynamic Code Injection

Some debuggers allow a user to enter arbitrary statements in an "immediate window" and execute the statements. The CLR debugger services support this scenario. Within reason, there are no restrictions on what code you can inject dynamically. For example, non-local "goto" statements are not allowed.

Dynamic Code Injection is implemented using a combination of Edit and Continue operations and Function Evaluation. The code to be injected is wrapped in a function and injected using Edit and Continue. The injected function is then evaluated. The wrapper function can be supplied with arguments that are declared to be ByRef so that side effects are immediate and permanent if desired.

3. Debugging Architecture

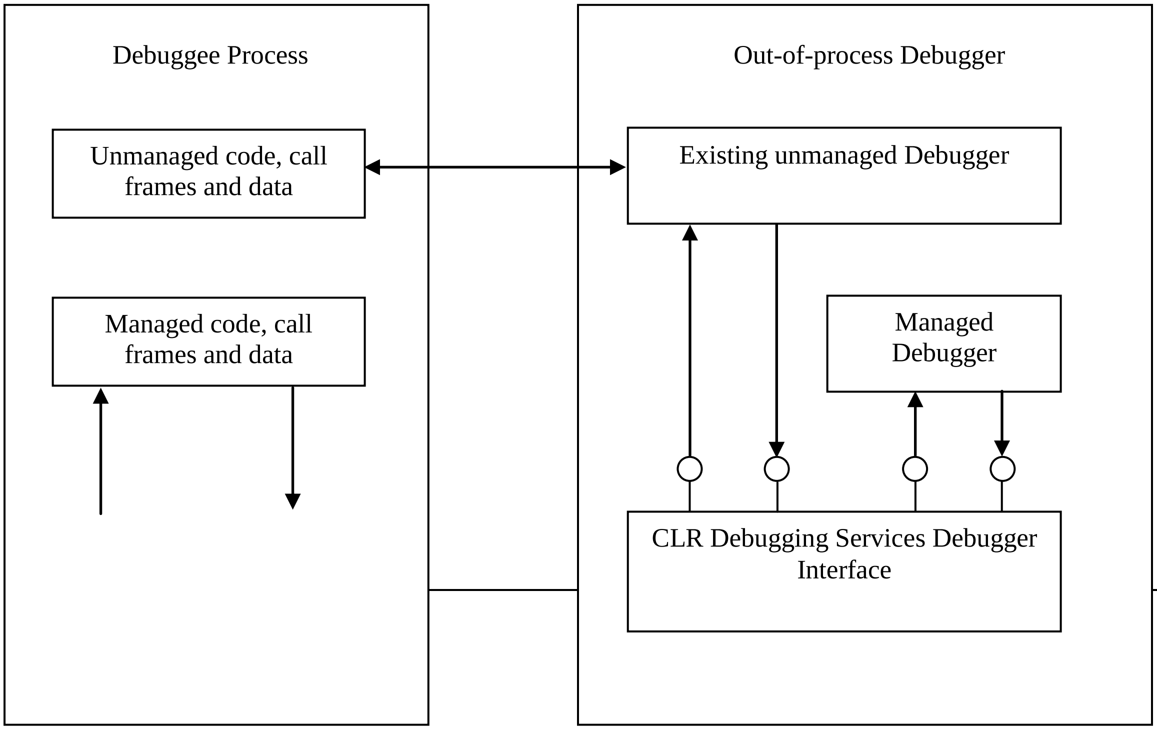
The CLR debugging services are designed to be used as if it were part of the operating system kernel. Today, when a program generates an exception the kernel gets involved to suspend execution of the process and pass the exception information to the debugger via the Win32 debugging API. The CLR debugging services serves the same purpose for managed code. When managed code generates an exception, the CLR debugging services will get involved to suspend the execution of the process and pass the exception information to the debugger. The sections below describe how and when the CLR debugging services get involved and what services they provide.

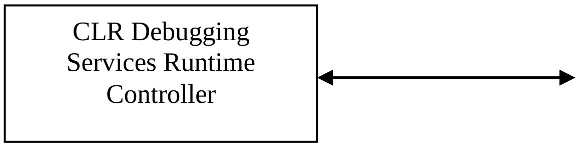
3.1 Process Architecture

One portion of the debugging services is implemented as a component that is always loaded into the same process with the program being debugged. This *Runtime Controller* component is responsible for communicating with the CLR and performing execution control and inspection of threads running managed code.

The other part of the code is implemented as a component that is loaded into a different process. This *Debugger Interface* component is responsible for communicating with the Runtime Controller on behalf of the debugger. It is also responsible for handling Win32 debugging events coming from the process being debugged, and either handling them or passing them on for use by an unmanaged code debugger. Finally, the Debugger Interface is the only part of the CLR debugging services with an exposed API.

The figure below illustrates where the different components of the CLR debugging services are located and how they interact with the CLR and the debugger.


Debuggee Process
Unmanaged code, call frames and data
Managed code, call frames and data
Out-of-process Debugger
Existing unmanaged Debugger
Managed Debugger
CLR Debugging Services Debugger Interface



3.2 Managed Code Debugger

It is possible to build a purely managed code debugger. The CLR debugging services allow such a debugger to attach to a process on-demand using a “soft attach” mechanism. A debugger that is soft attached to a process can subsequently detach from the process.

3.3 Thread Synchronization

The CLR debugging services have conflicting requirements when it comes to process architecture. On one hand, there are many compelling reasons for keeping the debugging logic in the same process as the program being debugged. For example, the data structures are complex and in many cases are manipulated by APIs rather than by a fixed memory layout. It is much easier to call the APIs directly rather than trying to decode the data structures from outside the process. Another example to keep the debugging logic in the same process is for better performance. There is no need for cross process communication in this case. Lastly, an important feature of CLR debugging is the ability to run user code in process with the debuggee, which obviously requires some cooperation with the debuggee process.

On the other hand, CLR debugging must coexist with unmanaged native code debugging. This can only be performed correctly from an outside process. Also, an out-of-process debugger is safer than an in-process debugger because interference between the debugger’s operation and the process being debugged is minimized.

Because of these conflicting requirements, the CLR debugging services combine some of each approach. The primary debugging interface is out-of-process and coexists with the native Win32 debugging services. However, the CLR debugging services add the ability to *synchronize* with the debuggee process so that it can safely run code in the user process. To perform this synchronization the services collaborate with the OS and the CLR to suspend all threads in the process at a place where they are not in the middle of an operation which leaves the runtime in an incoherent state. The debugger is then able to run code in a special thread which can examine the state of the runtime using the correct APIs, as well as call user code if the need arises.

When managed code executes a breakpoint instruction or generates an exception, the Runtime Controller is notified. This component will determine which threads are executing managed code and which threads are executing unmanaged code. Usually, threads that are running managed code will be allowed to continue executing until they reach a state where they may be safely suspended. This may include completing a GC in progress. Once the managed code threads have reached safe states, all threads will be suspended. The Debugger Interface then informs the debugger that a breakpoint or exception has been received.

When unmanaged code executes a breakpoint instruction or generates an exception, it is the Debugger Interface component which receives notification (through the Win32 debugging API). This notification is passed to an unmanaged debugger. If the debugger decides that it wants to perform synchronization (for instance, so managed code stack frames can be inspected), the Debugger Interface must first restart the stopped debuggee process, and then inform the Runtime Controller to perform the synchronization. The Debugger Interface is then notified when synchronization is complete. This synchronization is transparent to the unmanaged debugger.

The thread which generated the breakpoint instruction or exception must not be allowed to execute while all of this synchronization is going on. In order to facilitate this, the Debugger Interface “hijacks” the thread by placing a special exception filter in the thread’s filter chain. When the thread is restarted it will enter this filter which will place the thread under the Runtime Controller’s control. When it is later time to continue exception processing (or cancel the exception) the filter will return control to the thread's normal exception filter chain (or return the proper result to resume execution.).

In rare cases, the thread which generates the native exception may be holding crucial locks which need to be released before the runtime's synchronization can complete. (Typically these will be low-level library locks, for instance a lock on the malloc heap.) In such cases, the synchronization operation must timeout and the synchronization will fail. This will cause certain operations that require the synchronization to fail.

3.4 The In-Process Helper Thread

In order for the CLR debugging services to operate correctly, a single *debugger helper thread* is used within every CLR process. This helper thread is responsible for handling many of the inspection services provided by the API in addition to assisting with thread synchronization under certain circumstances.

3.5 Interactions with the Just in Time compilers

In order to allow a debugger to debug JIT-compiled code, the CLR debugging services must be able to map information from the MSIL version of the function to the native version of the function. This information includes sequence points in the code and local variable location information. Typically, this information takes extra resources to produce and keep around, so it is not produced unless the runtime is in debugging mode.

Also, JIT-compiled code can be highly optimized. Optimizations such as common sub-expression elimination, inline expansion of functions, loop unwinding, code hoisting, etc. can lead to loss of correlation between the MSIL code of a function and the native code which will be called to execute it. Thus, the JIT compiler's ability to correctly provide mapping information is severely impacted by these aggressive code optimization techniques. Therefore, when the runtime is run in debugging mode, the JIT compiler will not perform certain optimizations. This will allow debuggers to accurately determine the source line mapping and location of all local variables and arguments.

3.6 Debugging Modes

While we have tried to avoid introducing special modes for debugging, there are two cases where such modes are required.

The first case is when Edit and Continue functionality is desired. In this case the runtime actually operates differently to allow code to be later changed. This is because the layout of certain runtime data structures needs to be different to support Edit and Continue. Since this has a negative impact in performance, this mode should not be used unless Edit and Continue is desired. This mode is termed "Edit and Continue" mode.

The second case is needed so the JIT compiler can generate mapping information (see above discussion about JIT). Again, since this has a negative impact on performance (both from decreased code quality and increased effort and space to generate and remember mapping information), this mode should not be used unnecessarily. This mode is called simply "JIT Debugging Mode".

If a program is debugged while not in Edit and Continue mode, Edit and Continue functionality is not supported. If a program is debugged while not in debugging mode, most of the debugging features will still be supported; however only raw native code will be visible for JIT-compiled method frames – source and local variable mapping will be unavailable.

Both of these modes can be enabled programmatically through the CLR debugging services by a debugger which gains control of a process before the runtime has initialized itself. This is adequate for many purposes. However, a debugger attaching to a process which has already been running for a while (for instance, during Just-In-Time debugging) will be unable to activate these modes.

To help deal with these problems, a program can be run in Just-In-Time mode or Debugging mode independently of a debugger. Mechanisms for enabling debugging are described in the specification Enabling Profiling and Debugging.

JIT optimizations can make an application less debuggable. The CLR debugging services allow inspection of stack frames and inspection of local variables with JIT-compiled code that has been optimized. Stepping is supported but the stepping may be imprecise. A program can be run that instructs the JIT compiler to turn off all JIT optimizations causing the JIT compiler to produce “debuggable code.” For details see the specification Enabling Profiling and Debugging.