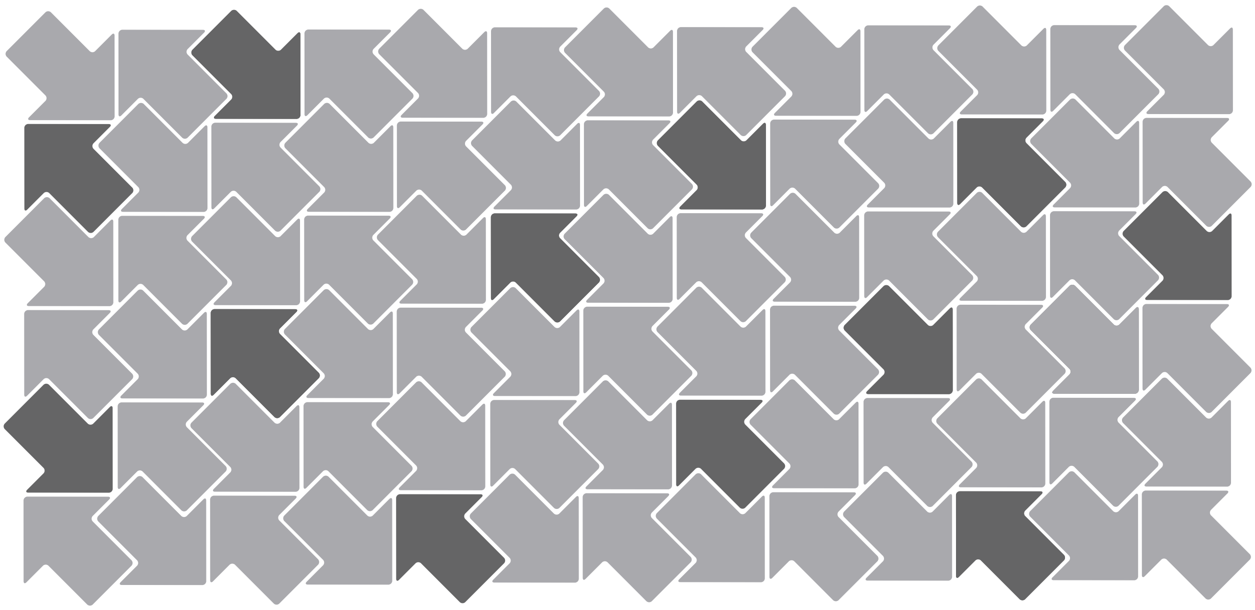
Visual Studio Integrated Virtual Debugger User's Manual

Workstation 6.0



Visual Studio Integrated Virtual Debugger User's Manual
Revision: 20070503
Item: WS-ENG-Q107-298

You can find the most up-to-date technical documentation on our Web site at

http://www.vmware.com/support/

The VMware Web site also provides the latest product updates.

If you have comments about this documentation, submit your feedback to:

docfeedback@vmware.com

© 2007 VMware, Inc. All rights reserved. Protected by one or more of U.S. Patent Nos. 6,397,242, 6,496,847, 6,704,925, 6,711,672, 6,725,289, 6,735,601, 6,785,886, 6,789,156, 6,795,966, 6,880,022, 6,944,699, 6,961,806, 6,961,941, 7,069,413, 7,082,598, 7,089,377, 7,111,086, 7,111,145, 7,117,481, 7,149, 843 and 7,155,558; patents pending.

VMware, the VMware "boxes" logo and design, Virtual SMP and VMotion are registered trademarks or trademarks of VMware, Inc. in the United States and/or other jurisdictions. All other marks and names mentioned herein may be trademarks of their respective companies.

VMware, Inc.
3145 Porter Drive
Palo Alto, CA 94304
www.vmware.com

Contents

Preface

This preface provides information about the *Visual Studio Integrated Virtual Debugger User's Manual* and links to VMware® technical support and educational resources.

This preface contains the following topics:

- "About This Book" on page 5
- "Technical Support and Education Resources" on page 6

About This Book

This manual, the *Visual Studio Integrated Virtual Debugger User's Manual*, provides information about setting up and using the Visual Studio Integrated Virtual Debugger.

Intended Audience

This book is intended for anyone who needs to set up or use the Visual Studio Integrated Virtual Debugger. The integrated virtual debugger is intended primarily for people who do software development, testing, and debugging in multiple Windows operating systems or computing environments.

Document Feedback

If you have comments about this documentation, submit your feedback to:

docfeedback@vmware.com

Conventions

Table P-2 illustrates the typographic conventions used in this manual.

Table P-1. Conventions Used in This Manual

Style	Elements
Blue (online only)	Cross-references and email addresses
Blue boldface (online only)	Links
Black boldface	User interface elements such as button names and menu items
`Monospace`	Commands, filenames, directories, and paths
`Monospace bold`	User input
Italic	Document titles, glossary terms, and occasional emphasis
< Name >	Variable and parameter names

Technical Support and Education Resources

The following sections describe the technical support resources available to you.

Self-Service Support

Use the VMware Technology Network (VMTN) for self-help tools and technical information:

- Product information – http://www.vmware.com/products/
- Technology information – http://www.vmware.com/vcommunity/technology
- Documentation – http://www.vmware.com/support/pubs
- VMTN Knowledge Base – http://www.vmware.com/support/kb
- Discussion forums – http://www.vmware.com/community
- User groups – http://www.vmware.com/vcommunity/usergroups.html

For more information about the VMware Technology Network, go to http://www.vmtn.net.

Online and Telephone Support

Use online support to submit technical support requests, view your product and contract information, and register your products. Go to http://www.vmware.com/support.

Customers with appropriate support contracts should use telephone support for the fastest response on priority 1 issues. Go to http://www.vmware.com/support/phone_support.html.

Support Offerings

Find out how VMware support offerings can help meet your business needs. Go to http://www.vmware.com/support/services.

VMware Education Services

VMware courses offer extensive hands-on labs, case study examples, and course materials designed to be used as on-the-job reference tools. For more information about VMware Education Services, go to http://mylearn1.vmware.com/mgrreg/index.cfm.

Overview of the Visual Studio Integrated Virtual Debugger

1

The Visual Studio Integrated Virtual Debugger provides a configurable interface between Visual Studio and virtual machines, making it easy to develop and debug applications that run in multiple Windows operating system environments on a single PC. Debugging your applications in virtual machines enables you to reproduce and record errors while maintaining the integrity of the host machine. You can perform typical debugging tasks such as pausing at breakpoints, stepping through code, and viewing and modifying the state of your application, all without impacting the host environment.

You can manage configuration settings for each virtual machine in which you want to execute and debug applications. Virtual machine configuration properties, which you set in the Visual Studio Integrated Virtual Debugger configuration pages, determine which virtual machine to run the application in and how the application is executed. Using these configurations, you can:

- Start an application debugging session in a virtual machine.
- Start an application in a virtual machine without debugging.
- Start a debugging session that attaches to a process already running in a virtual machine.

Once configured, the integrated virtual debugger finds the virtual machine, powers it on if necessary, sets up the environment based on your configuration settings, and starts or attaches to the application.

This chapter lists the configuration options for starting an application in a virtual machine and for attaching to a process already running in a virtual machine. It contains the following sections:

- "Configuration Options When Starting an Application in a Virtual Machine" on page 7
- "Configuration Options When Attaching to a Process Running in a Virtual Machine" on page 8

For information on installation and other requirements for the Visual Studio Integrated Virtual Debugger, see Chapter 2, "Setting Up the Visual Studio Integrated Virtual Debugger Environment," on page 9.

For information on how to manage virtual machine configurations and how to start applications in virtual machines, see Chapter 3, "Using the Visual Studio Integrated Virtual Debugger," on page 15.

Configuration Options When Starting an Application in a Virtual Machine

To configure how to start an application (with or without debugging) in a virtual machine, you can specify the following settings in the Visual Studio Integrated Virtual Debugger configuration pages:

- The command to be executed by Visual Studio in the guest operating system.
- The name of the virtual machine (`.vmx` configuration file).
- Whether to run the command as a shared path on the host or as a guest path.
- The location of the Remote Debug Monitor on the host.
- The name of the Remote Debug Monitor on the guest.

You can specify the following additional settings when you start debugging an application in a virtual machine, but not when you start an application without debugging:

- (Optional) The location of folders to be shared between the host and the guest.
- (Optional) Actions to perform before starting an application in a virtual machine, including:
  - Copying files or folders from the host to the virtual machine.
  - Reverting the virtual machine to the parent snapshot.
  - Running specified pre-execution commands in the guest. For example, if you must register new DLLs in the virtual machine each time the program is recompiled, you can create a DLL registration script and specify that it must be run during setup.
- (Optional) Actions to perform after an application in a virtual machine is terminated, including:
  - Running specified post-execution commands (for example, to perform cleanup tasks) on the guest.
  - Setting the virtual machine state to:
    - No operation (remain powered on, no shutdown action)
    - Powered off
    - The parent snapshot
    - Suspended

Configuration Options When Attaching to a Process Running in a Virtual Machine

To configure a debugging session that attaches to a process already running in a virtual machine, you can specify the following in the Attach to Process dialog box:

- The name of the virtual machine (`.vmx` configuration file).
- The location of the Remote Debug Monitor on the host.
- The name of the Remote Debug Monitor on the guest.

Setting Up the Visual Studio Integrated Virtual Debugger Environment 2

Review the requirements and recommendations in this chapter before following the instructions in the *Workstation User's Manual* to install the Visual Studio Integrated Virtual Debugger as an optional component of Workstation 6. The Visual Studio Integrated Virtual Debugger can be installed on any supported Windows host system that is running Workstation 6 and has a supported version of Visual Studio installed.

When you install the Visual Studio Integrated Virtual Debugger:

- The associated DLLs are placed in the `\Program Files\VMware\VMware Workstation\Visual Studio Integrated Debugger` and `\Program Files\VMware\VMware VIX\ws-2\32bit` directories.
- When you restart Visual Studio, the integrated virtual debugger is loaded and the **VMware** menu and toolbar become available.
- A preference file, `vsid-prefs.xml`, is created in the `\Documents and Settings\`<user_name>`\Application Data\VMware` directory. Do not edit this file directly. It is updated when you make changes in the integrated virtual debugger configuration pages.
- A file, <project_name>`.idc`, is created for each project in same directory as the project file when a project of a type supported by the integrated virtual debugger is loaded in Visual Studio.
- A log file, `vmware-vsid-`<integer>`.log`, is created in the `\Documents and Settings\`<user_name>`\Local Settings\Temp` directory. You can choose **VMware>About VMware Virtual Debugger** to view the log file name. This log file contains informational and error messages about the actions of the integrated virtual debugger.

You can debug in multiple virtual machines simultaneously. You can also debug multiple sessions in a single virtual machine. However, you cannot debug on a local or physically remote machine and in a virtual machine at the same time.

This chapter contains the following sections:

Microsoft Visual Studio Requirements and Recommendations

This section includes requirements and configuration recommendations for Visual Studio.

Visual Studio must be running on the same system as Workstation 6.

Supported Versions of Visual Studio

Only Visual Studio 2005 Professional and Team Systems editions are supported. These versions of Visual Studio allow remote debugging on Windows systems, with the exceptions of Windows NT and Windows

Vista Starter Edition. The Visual Studio Integrated Virtual Debugger uses the features of the Remote Debug Monitor (`msvsmon.exe`) to communicate with the guest operating system.

VMware recommends that you install Visual Studio 2005 SP1. For more information, see http://msdn2.microsoft.com/en-us/vstudio/bb265237.aspx

For important information about running Visual Studio 2005 on Windows Vista, see http://msdn2.microsoft.com/en-us/vstudio/aa972193.aspx

Running the Visual Studio Integrated Virtual Debugger on Windows Vista Starter Edition is not supported. For information about issues running Visual Studio on Windows Vista Starter Edition, see http://msdn2.microsoft.com/en-us/vstudio/aa964140.aspx#question46.

Supported Languages

The C/C++ (Native and Managed), C#, and Visual Basic languages are supported.

Configuring the Runtime Library Setting for C++ Applications

When you debug on a physically remote machine or in a virtual machine, the application might not start if the runtime library setting is set to certain values. If you encounter this problem, change the C++ runtime library setting.

To update the Visual Studio runtime library setting

1 Choose **Project >Properties**.

2 Expand **Configuration Properties>C/C++** and select **Code Generation**.

3 Set **Code Generation** to **Runtime Library property to Multi-threaded (/MT)** or **Multi-threaded Debug (/MTd)**.

For additional information, see http://msdn2.microsoft.com/en-us/library/ms235624.aspx.

Host System Requirements

The Visual Studio Integrated Virtual Debugger can run on most Windows host operating systems supported by Workstation 6 that have a supported version of Visual Studio installed. On Windows Server 2003, only Enterprise Edition SP1 and R2 are supported.

If remote debugging is not working on a Windows Vista host, try the following:

- Manually configure the firewall to allow traffic from Visual Studio.
- Run Visual Studio with Administrator permissions. For more information, see http://msdn2.microsoft.com/en-us/vstudio/aa972193.aspx.

Virtual Machine Requirements and Recommendations

This section includes requirements and configuration recommendations for virtual machines.

Guest Operating System Support

The Visual Studio Integrated Virtual Debugger is supported on any Workstation 6 virtual machine that is running a supported Windows guest operating system, with the exceptions of Windows NT, Windows Me, Windows 98, Windows 95, Windows for Workgroups, Windows 3.1, Windows XP Home Edition, and Windows Vista Starter Edition.

Make sure that the version of VMware Tools on the guest operating system matches the version of Workstation 6 (which the Visual Studio Integrated Virtual Debugger is a component of) on the host.

Configuring the Network

Set up the virtual machine network as **Bridged** or **Host-only**.

On Windows XP, in the guest system **Control Panel>Administrative Tools>Local Security Policy>Local Policies>Security Options** page, set the policy **Network access: Sharing and security model for local accounts** to **Classic - local users authenticated as themselves**.

On Windows Vista, **Classic - local users authenticated as themselves** is the default value for this policy. To verify that this policy is set correctly, follow the same steps as for Windows XP.

NOTE It is not possible to view this policy on Windows Vista Home Premium and Vista Home Basic.

Configuring the Firewall on Windows XP SP2 Virtual Machines

Windows XP SP2 systems have the firewall enabled by default. To debug in a virtual machine with Windows XP SP2, you must disable the firewall or configure it appropriately. VMware recommends disabling the firewall. Virtual machines are protected behind the host firewall.

For information on setting up remote debugging in Visual Studio using Windows XP SP2 with the firewall enabled, see:

- http://msdn.microsoft.com/security/productinfo/xpsp2/default.aspx?pull=/library/en-us/dnwxp/html/xpsp2remotedebug.asp
- http://support.microsoft.com/default.aspx?scid=kb;%5BLN%5D;833977#2020

Configuring User Accounts

Log in to the guest operating system with an Administrator account. Use the same local or domain user account on the host machine to log in to the guest operating system.

NOTE The user name, password, and domain name (if not local on both systems) must match on the host and the guest. Otherwise the Remote Debug Monitor on the guest will not be able to communicate with the Visual Studio debugger on the host.

For additional information on setting up Windows user accounts for remote debugging, see http://msdn2.microsoft.com/en-us/library/ms164725.aspx.

NOTE Communication between Visual Studio and the guest operating system is not initiated until the virtual machine is powered on and the configured user is logged in. This user runs the Remote Debug Monitor on the guest, which in turn communicates with the Visual Studio debugger on the host.

To prevent a time delay, power on the virtual machine and log in to the guest operating system before debugging in a virtual machine. You can set up automatic login to bypass the login screen when the guest is booting, as described next.

Setting the Password Policy

Windows has a default security feature that helps protect users with blank passwords from network-based attacks. Users who do not password-protect their accounts can log in only at their physical computer console: the monitor, keyboard, and mouse that is physically connected to their computer. This restriction applies only to local user accounts, not to domain user accounts.

For information on how to disable blank password restrictions, see http://support.microsoft.com/?id=303846.

Suppressing Security Prompts

Running an application from a network share triggers a security prompt every time the file is accessed. VMware recommends that you turn off security prompts on the guest operating system.

To turn off security prompts on the guest system

1. In Internet Explorer, choose **Tools>Internet Options>Security>Local Intranet**, and click **Sites**.
2. Click **Advanced**, and add a new Web site: `file://*..host`

Alternatively, you can edit the registry key directly.

To turn off security prompts by editing the registry key

1. Open the registry.
2. Add a new key, `.host`, under `HKCU\Software\Microsoft\Windows\CurrentVersion\InternetSettings\ZoneMap\Domains`.
3. In the `.host` key, create a new **DWORD Value** called `file` and set its value to `1`.

Installing the Microsoft .NET Framework to Support Managed Applications

To debug managed C++, C#, and Visual Basic applications, which use the Common Language Runtime, you must install the Microsoft .NET Framework version 2.0 or higher on the guest operating system.

Using Unique Virtual Machine Computer Names

Verify that the computer names are unique on all virtual machines, otherwise the Visual Studio Integrated Virtual Debugger cannot find the appropriate virtual machine on the network.

To rename a computer

1. On the guest system, choose **Start>Control Panel>System**.
2. Select the **Computer Name** tab.
3. Click **Change**.
4. Type a unique name, and click **OK**.

Installing and Starting the Remote Debug Monitor Manually on Windows 98 Guest Systems

To enable debugging in a virtual machine, the Visual Studio Integrated Virtual Debugger shares the host folder that contains the Remote Debug Monitor, and runs that Remote Debug Monitor on the guest. On Windows 98, it is not possible to run the Remote Debug Monitor (or any executable) from a shared folder. An attempt to do so generates the error: `The remote debugger is not properly installed. On a Windows ME or Windows 98 computer, the debugger cannot be run off a file share. Run the remote debugger setup.`

Instead, you must manually install and start the Remote Debug Monitor executable, `msvsmon.exe`, on the guest operating system before starting a debugging session. You can copy `msvsmon.exe` to the guest before starting the debug session, create a mapping to a network share with the host where `msvsmon.exe` is located, or install `msvsmon.exe` from the Visual Studio 2005 installation CD.

NOTE Once the debugging session starts, click **Yes** when prompted to use the existing instance of the Remote Debug Monitor.

Due to this shared folder limitation, you must also:

- Set **Run Command As** to **a guest path**. This property, which indicates how the command being executed by the debugger is run, is described in "Setting General Properties" on page 16.
- Leave **Shared Directories** unset, because directories cannot be shared between the host and the guest. This property is described in "Setting Virtual Machine Properties" on page 17.

Troubleshooting Tips

This section contains additional information that enables you to use the Visual Studio Integrated Virtual Debugger successfully.

Changing Shortcut Keys

If you change the shortcut keys for `VMDebugger` commands (in **Tools>Options>Keyboard**), the tooltips for the VMware menu and toolbar will not reflect the changes until you restart Visual Studio.

Reinstalling VMware Tools If the Debugging Session Does Not Start

If the debugging session fails to start and the last message in the VMware output window (and log file) is `Waiting for VMware Tools to start`, check whether the guest system has the latest VMware Tools installed and running. If not, upgrade to the latest version of VMware Tools.

Exiting Visual Studio Before Powering Off a Virtual Machine

If you attempt to exit Visual Studio after starting a debugging session but before logging in or running VMware Tools on the guest operating system, Visual Studio will not exit until the virtual machine is powered off or the user is logged in to the guest operating system.

Unloading the VMDebugger Add-in

To permanently uninstall the Visual Studio Integrated Virtual Debugger, run the Workstation installation program, select **Modify** on the Program Maintenance page, deselect **Visual Studio PlugIn** in the **Custom** setup, and continue through the installation wizard.

NOTE Deselecting **Start** in **Tools>Add-In Manager** does not prevent the Visual Studio Integrated Virtual Debugger Add-in from loading.

Cleaning Up After a Crash

If you try to run the debugger locally after a debugging session in a virtual machine crashes or freezes, you might get a Visual Studio error that indicates that the remote server cannot be found. To reset Visual Studio to debug locally:

- In C++, choose **Project Property Pages>Debugging** and set the **Debugger to Launch** property to `Local Windows Debugger`. Set the **Command** property to either an empty string or the correct local path.
- In C# and VB, choose **Project Property Pages>Debug**. Make sure **Start project** is selected and **Use remote machine** is deselected.

Optionally, remove any shared folders that were used to run the debug command and the Remote Debug Monitor. Shared folders are usually removed at the end of a debugging session, but they might not be removed in the following circumstances:

- If the debugging session causes a crash.
- If the virtual machine is powered off while the debugging session is still running.

These shared folders are typically reused when another debugging session is started, so this cleanup is not required.

Using the Visual Studio Integrated Virtual Debugger 3

The Visual Studio Integrated Virtual Debugger enables you to:

- Manage configuration settings for application execution and debugging in virtual machines.
- Start an application debugging session in a virtual machine.
- Start an application in a virtual machine without debugging.
- Start a debugging session that attaches to a process already running in a virtual machine.

This chapter contains the following sections:

The Visual Studio Integrated Virtual Debugger uses the features of the Remote Debug Monitor (`msvsmon.exe`) to communicate with the guest operating system. Chapter 2, "Setting Up the Visual Studio Integrated Virtual Debugger Environment," on page 9 describes the Visual Studio, host system, and virtual machine requirements to perform remote debugging.

Managing Configurations

Before you can start or debug applications in a virtual machine, you must create or modify virtual machine configurations and set configuration properties. The Default configuration initially includes the default values for all properties that have them.

Choose **VMware>Options** to manage configurations. You can create, rename, and remove configurations as described in this section, and you can set and modify configuration properties for existing configurations as described in "Setting Configuration Properties" on page 16.

The configuration selected in the **Configuration** drop-down menu is the one being edited in the configuration pages, while the configuration selected in the **Active Configuration** drop-down menu is the one used when you choose **VMware>Start** or **VMware>Start Without Debugging**.

Creating Configurations

This section describes how to create a new configuration.

To create a new configuration

1 Choose **VMware>Options**.

2 Click the **New** icon next to the **Configuration** drop-down menu.

3 In the New Configuration page, type a name for the new configuration.

4 Choose a configuration to copy settings from.

   The default selection is **<Default>**, which includes the default values for all properties that have them.

5 Click **OK**.

The new configuration is created and listed as the active configuration in the **Configuration** and **Active Configuration** drop-down menus. Next, edit the configuration properties, as described in "Setting Configuration Properties" on page 16.

Renaming Configurations

You can rename an existing configuration.

To rename a configuration

1 Choose **VMware>Options**.

2 Choose the configuration you want to rename from the **Configuration** drop-down menu, and click the **Edit** icon.

3 In the Edit Configuration page, select the configuration you want to rename, and click **Rename**.

4 Type the new name over the existing name, and press **Enter**.

5 At the confirmation prompt, click **Yes**.

6 Click **Close**.

The renamed configuration is listed as the active configuration in the **Configuration** drop-down menu. You can edit its configuration properties, as described in "Setting Configuration Properties" on page 16.

Removing Configurations

You can remove an existing configuration.

To remove a configuration

1 Choose **VMware>Options**.

2 Choose the name of the configuration you want to delete from the **Configuration** drop-down menu, and click the **Edit** icon.

3 In the Edit Configuration page, select the configuration you want to delete, and click **Remove**.

4 At the confirmation prompt, click **Yes**.

5 Click **Close**.

The configuration is removed from the **Configuration** drop-down menu.

Setting Configuration Properties

You can edit configuration properties for a specific configuration by choosing the configuration name from the **Configuration** drop-down menu. You can also edit configuration properties for all configurations by choosing **All Configurations** from the **Configuration** drop-down menu.

The Default configuration initially includes the default values for all properties that have them.

Setting General Properties

General properties include:

- The command to be executed by Visual Studio in the guest operating system.
- How the command is run: as a path on the host in a shared folder or as a path on the guest.

- The location of the Remote Debug Monitor on the host.
- The name of the Remote Debug Monitor on the guest.

To set general properties

1. Choose **VMware>Options**, and select **General** in the left pane.
2. Set **Command** to the command to be executed by the debugger in the guest system. Click **Browse** to select a path to the executable on the host file system.

   The command directory is automatically shared between the host and the guest.
3. Set **Run Command As** to indicate how the debug command is run: either as **a host path through a shared folder** or **a guest path**.

   When **a host path through a shared folder** is selected, the folder where the command is located is shared before the debugging session is started. The command is executed from the shared folder, and when the debugging session ends, the folder is no longer shared. The name of the shared folder is `\\.host\Shared Folders\$(ProjectName)`<random_number>.

   When **a guest path** is selected, the command is executed from the specified path on the guest.

   The default is **a host path through a shared folder**.
4. Set **Remote Debug Monitor** to the location of the Remote Debug Monitor on the host.

   The default is the Visual Studio installed path, typically:

   ```
   \Program Files\Microsoft Visual Studio 8\Common7\IDE\Remote Debugger\x86\msvsmon.exe
   ```

   Use the default remote debug monitor if you are debugging a 32-bit process in a 32-bit virtual machine.

   If you want to debug a 32-bit process in a 64-bit virtual machine, use the Remote Debug Monitor:

   ```
   \Program Files (x86)\Microsoft Visual Studio 8\Common7\IDE\Remote Debugger\x86\msvsmon.exe
   ```

   If you want to debug a 64-bit process in a 64-bit virtual machine, use the 64-bit Remote Debug Monitor:

   ```
   \Program Files\Microsoft Visual Studio 8\Common7\IDE\Remote Debugger\x64\msvsmon.exe
   ```
5. Type a name for the Remote Debug Monitor on the guest.

   The default name is <user name>.

   If a Remote Debug Monitor is already running on the guest, when the new connection is made between the Remote Debug Monitor on the guest and the Visual Studio debugger on the host, you are prompted to choose whether to connect to the one that is running or start another one with a different name.

Setting Virtual Machine Properties

Virtual machine properties include:

- The path to the virtual machine file (`.vmx` file).
- (Optional) The location of any directories shared between the host and the guest.

To set virtual machine properties

1. Choose **VMware>Options**, and select **Virtual Machine** in the left pane.
2. Set **Virtual Machine** to the path to the virtual machine file (`.vmx` file). Click **Browse** to select from `.vmx` files on the system.
3. (Optional) Set **Shared Folders** to a semicolon-delimited list of paired folder names in the form <shared_name>=<host_folder_name>. Click **Browse** to enter share names and folder names using a dialog box.

(Optional) Setting Pre-Debug Event Properties

Pre-Debug Event properties determine which actions are performed before the debug command is run. All of these settings are optional.

To configure actions to perform before beginning debugging

1. Choose **VMware>Options**, and select **Pre-Debug Event** in the left pane.
2. Set **Revert to Parent Snapshot** to **Yes** or **No**.

   If set to **Yes**, the virtual machine reverts to its parent snapshot when the debugging session is started.

   The default is **No**.
3. Set **Copy Files** to a semicolon-delimited list of paired file or directory names that are copied from the host to the guest machine in the form <host_machine_file/folder>=<virtual_machine_file/folder>. Click **Browse** to select from files on the system.

   All specified files are copied before any pre-debugging commands are executed.
4. Set **Command Line** to one or more semicolon-delimited commands that are run after files are copied (as described in the preceding step) and before the debugging session starts. Click **Browse** to enter commands using a dialog box.

(Optional) Setting Post-Debug Event Properties

Post-Debug Event properties determine which actions are performed after the debug command is terminated. All of these settings are optional.

To configure actions to perform after debugging has occurred

1. Choose **VMware>Options**, and select **Post-Debug Event** in the left pane.
2. Set **Command Line** to one or more semicolon-delimited commands that are run after the debugging session ends. Click **Browse** to enter commands using a dialog box.
3. Set **Termination Mode** to:
   - No operation (default)
   - Power off
   - Revert to parent snapshot
   - Suspend

Starting a Debugging Session in a Virtual Machine

You can debug an application in any configured virtual machine.

To start a debugging session in a virtual machine

1. Choose **VMware>Start**.

   NOTE You must log in to the guest system manually before the application is started. For additional information, see "Configuring User Accounts" on page 11.

   The application is started in the virtual machine.
2. Perform debugging tasks as you would from the **Debug>Start Debugging** Visual Studio menu.

If you want to kill the processes associated with the debugging session on the guest system and restart debugging, choose **VMware>Restart**.

Starting a Session Without Debugging in a Virtual Machine

You can start an application in any configured virtual machine without debugging.

When you start an application without debugging, the Visual Studio Integrated Virtual Debugger

1 Powers on the virtual machine if necessary.

2 Shares the folder to the executable.

3 Runs the executable.

4 Removes the shared folder when the executable terminates.

When you start an application without debugging, the integrated virtual debugger does not execute pre-debug or post-debug operations, share additional directories, or start the Remote Debug Monitor on the guest system.

To start an application in a virtual machine without debugging

Choose **VMware>Start Without Debugging**.

NOTE You must log in to the guest system manually before you can run the application. For additional information, see “Configuring User Accounts” on page 11.

The application is started in the virtual machine.

Attaching the Debugger to a Process Running in a Virtual Machine

You can debug an application that is already running in a virtual machine.

To attach the debugger to a running process

1 Choose **VMware>Attach to Process**.

The Attach to Process page is displayed.

2 Choose the virtual machine on which to view running processes from the **Running Virtual Machines** drop-down menu.

Only virtual machines that are powered on appear in the drop-down menu.

3 Set **Remote Debug Monitor** to the location of the Remote Debug Monitor on the host.

The default is the Visual Studio installed path, typically:

```
\Program Files\Microsoft Visual Studio 8\Common7\IDE\Remote Debugger\x86\msvsmon.exe
```

Use the default remote debug monitor if you are debugging a 32-bit process in a 32-bit virtual machine.

If you want to debug a 32-bit process in a 64-bit virtual machine, use the Remote Debug Monitor:

```
\Program Files (x86)\Microsoft Visual Studio 8\Common7\IDE\Remote Debugger\x86\msvsmon.exe
```

If you want to debug a 64-bit process in a 64-bit virtual machine, use the 64-bit Remote Debug Monitor:

```
\Program Files\Microsoft Visual Studio 8\Common7\IDE\Remote Debugger\x64\msvsmon.exe
```

4 Type a name for the Remote Debug Monitor on the guest.

The default name is `VMDebug`.

NOTE If a Remote Debug Monitor is already running on the guest, you can start another one with a different name or use one that is already running.

---

5 Choose the process you want to attach to from the list of available processes, and click **Attach**.

If you want to refresh the list of running processes, click **Refresh**.

Index