Integrated Virtual Debugger for Eclipse Developer's Guide

VMware Workstation 8.0

This document supports the version of each product listed and supports all subsequent versions until the document is replaced by a new edition. To check for more recent editions of this document, see http://www.vmware.com/support/pubs.

EN-000172-01

vmware®

You can find the most up-to-date technical documentation on the VMware 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

Contents

About This Book

The *Integrated Virtual Debugger for Eclipse Developer's Guide* provides information on how to install the Integrated Virtual Debugger for Eclipse. The guide also provides information on how to manage launch configurations to debug Java or C and C++ applications in multiple virtual machines simultaneously or multiple sessions in a single virtual machine.

Intended Audience

This book is intended for anyone who wants to debug in a virtual machine. The information in this book is written for experienced Windows or Linux developers and QA Engineers who are familiar with virtual machine technology and the debugging process.

Document Feedback

VMware welcomes your suggestions for improving our documentation. If you have comments, send your feedback to docfeedback@vmware.com.

Technical Support and Education Resources

The following sections describe the technical support resources available to you. To access the current version of this book and other books, go to http://www.vmware.com/support/pubs.

Online and Telephone Support

To 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

To find out how VMware support offerings can help meet your business needs, go to http://www.vmware.com/support/services.

VMware Professional Services

VMware Education Services courses offer extensive hands-on labs, case study examples, and course materials designed to be used as on-the-job reference tools. Courses are available onsite, in the classroom, and live online. For onsite pilot programs and implementation best practices, VMware Consulting Services provides offerings to help you assess, plan, build, and manage your virtual environment. To access information about education classes, certification programs, and consulting services, go to http://www.vmware.com/services.

Using the Integrated Virtual Debugger for Eclipse

1

The Integrated Virtual Debugger for Eclipse provides a configurable interface between Eclipse and virtual machines making it easy to develop and debug, on a single PC, applications that run on multiple operating systems. Debugging your applications in virtual machines enables you to reproduce and record errors while maintaining the integrity of the host machine. This chapter contains the following sections:

-
-
-

Debugging Environment of the Integrated Virtual Debugger

The Integrated Virtual Debugger lets you debug Java or C++ applications in a virtual machine. It is often necessary to debug an application in many different environments. With the Integrated Virtual Debugger you can debug a Java or C++ application on different operating systems and different JVMs. For example, you can also debug using different versions of Windows, service packs, DLLs installed, and so on.

Using virtual machines for debugging provides a convenient alternative to debugging on physical machines because you can debug an application on different setups. The Integrated Virtual Debugger helps manage your virtual machines. The Integrated Virtual Debugger can power guest virtual machines on and off, start the application in its virtual machine, attach the debugger to the application, and control the application.

The typical debugging tasks such as pausing at breakpoints, stepping through code, and viewing and modifying the state of your application, can be performed without impacting the host environment.

The Integrated Virtual Debugger also enables you to:

- Manage launch 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.

Using Eclipse launch configurations, you can choose a virtual machine in which to run your application and determine how the application is executed. When 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.

To configure how an application is started in a virtual machine, you can specify:

- Name of the virtual machine (`.vmx` configuration file).
- Account credentials for guest console.
- (Optional) Path to the Java Virtual Machine (JVM) on the guest system.
- (Optional) Locations of folders to be shared between the host and the guest.

- (Optional) Actions to take before launching an application from Eclipse, including:
    - Revert to the most recent snapshot.
    - Run specified pre-execution commands.
- (Optional) Actions to take after an application launched from Eclipse is terminated, including:
    - Run specified post-execution commands (for example, to perform cleanup tasks).
    - Set the virtual machine state to:
        - Suspended (default)
        - Revert to the most recent snapshot
        - Powered off

Installation Requirements of the Integrated Virtual Debugger for Eclipse

The Integrated Virtual Debugger is supported for Workstation 7.0.

Host System Requirements

The Integrated Virtual Debugger for Eclipse can run on any supported host operating system that is running Workstation 7.0 and has Eclipse installed. Eclipse must be running on the same host system as Workstation 7.0. For information about Eclipse versions, see "Eclipse Requirements" on page 9.

Supported Host Operating Systems

The Integrated Virtual Debugger supports the following Windows 32-bit, Windows 64-bit, Linux 32-bit, and Linux 64-bit host operating systems.

Table 1-1. Windows and Linux Host Operating Systems

Operating System	Edition
Windows 32-bit	Windows 7 Ultimate Edition
	Windows 7 Enterprise Edition
	Windows 7 Professional
	Windows 7 Home Premium
	Windows 7 Home Basic
	Windows Vista Enterprise Edition, SP1
	Windows Vista Business Edition, SP1
	Windows Vista Home Basic and Premium Editions, SP1
	Windows Vista Ultimate Edition, SP1, SP3
	Listed versions are also supported with no service pack.
	Windows XP Home Edition with SP2 or later service pack
	Windows XP Professional with SP2 or later service pack
Windows 64-bit	Windows 7 Ultimate Edition
	Windows 7 Enterprise Edition
	Windows 7 Professional
	Windows 7 Home Premium
	Windows 7 Home Basic
	Windows XP Professional with SP1 or later service pack
Linux 32 and 64-bit	Red Hat Enterprise Linux WS 4.5 (Beta, formerly called 4.0 Update 5)
	Red Hat Enterprise Linux AS 4.0, updates 1, 2, 3, 4
	Red Hat Enterprise Linux ES 4.0, updates 1, 2, 3, 4
	Red Hat Enterprise Linux WS 4.0, updates 1, 2, 3, 4
	Red Hat Linux 9.0 — stock 2.4.20-8, upgrade 2.4.20-20.9
	Ubuntu Linux 8.04, 8.10, and 9.04
	Other Linux distributions might work, but were not tested.

Supported JRE Versions

The host system must be running a Java runtime environment (JRE) meeting Java 2 Platform Standard Edition (J2SE) 5.0 or higher specifications. J2SE consists of the JRE and developer tools for compiling, debugging, and running applications written in the Java language.

NOTE Eclipse displays the error message `unable to load class` if an unsupported version of J2SE is being used on the host system.

Eclipse Requirements

For debugging Java applications, the Integrated Virtual Debugger supports Eclipse 3.2, 3.3, 3.3.x, 3.4, and 3.4.x. Eclipse 3.5 might work, but was not tested. On Windows Vista hosts, you must have Eclipse 3.2.2 or higher installed. Only the Java language is supported.

NOTE You cannot have GNU Compiler for the Java programming language (GCJ) or GNU Interpreter for Java (GIJ) installed on either the host or guest operating system.

To debug C and C++ applications with the Integrated Virtual Debugger, you must have Eclipse Classic 3.4.x and higher and Java 6 installed on the host. On Windows hosts, you must have MinGW with gdbserver 6.6 installed. The MinGW with gdbserver 6.6 download is available from the sourceforge.net Web site.

Installing and Uninstalling the Integrated Virtual Debugger for Eclipse

You must install the appropriate version of Eclipse on the host. For information about the required Eclipse versions to debug Java or C and C++ applications, see "Eclipse Requirements" on page 9.

Before you begin installing Workstation, make sure that you have the prerequisites listed in "Installation Requirements of the Integrated Virtual Debugger for Eclipse" on page 8.

When you install the latest version of Workstation with Eclipse present, the Workstation installer installs the Integrated Virtual Debugger for Eclipse. For information about how to install Workstation, see the *Workstation User's Manual* on the VMware Web site. If you install Eclipse after you install Workstation, run the Workstation installer again and select the **Modify** option to install the Integrated Virtual Debugger for Eclipse.

When you install the Integrated Virtual Debugger for Eclipse:

- For the JRE, the Integrated Virtual Debugger plug-in, `ivd.jar`, Foundry Java bindings, and the `plugin.xml` launch configuration file are placed in the `com.vmware.bfg_1.0.0` subdirectory of the Eclipse plug-in directory.

  After you restart Eclipse, the **Debug** menu includes the new launch configuration types **VMware attach to application** and **VMware execute Java application**. These launch configuration types have a **VMware** tab that enables you to configure virtual machine settings.

- For C and C++ debugging, the Integrated Virtual Debugger plug-in installs in the following directories:

  ```
  <Eclipse_Features_Directory>/com.vmware.remotedebug.ccpp.feature_1.0.0/
  <Eclipse_Plugin_Directory>/com.vmware.remotedebug.ccpp_1.0.0.jar
  <Eclipse_Plugin_Directory>/com.vmware.vixapi_1.0.0.jar
  <Eclipse_Plugin_Directory>/com.vmware.vixservice.connectorservice_1.0.0.jar
  <Eclipse_Plugin_Directory>/com.vmware.vixservice.services_1.0.0.jar
  <Eclipse_Plugin_Directory>/com.vmware.vixservice.subsystems_1.0.0.jar
  <Eclipse_Plugin_Directory>/com.vmware.vixservice.ui_1.0.0.jar
  ```

To uninstall the Integrated Virtual Debugger for Eclipse uninstall Workstation. For more information about how to uninstall Workstation, see the *Workstation User's Manual* on the VMware Web site.

VMware Tools Requirements

Make sure that the version of VMware Tools on the debugged guest operating system matches the version of Workstation 7.0 (of which the Integrated Virtual Debugger is a component) on the host.

Remote Debugging in Java Environment

2

You can use remote debugging to debug a Java application from inside a virtual machine that has a different guest operating system from the host. This chapter includes the following sections:

- "Virtual Machine Requirements for Debugging Java Applications" on page 11
- "Managing Virtual Machine Launch Configurations" on page 13
- "Running and Debugging Applications in Virtual Machines" on page 14

Virtual Machine Requirements for Debugging Java Applications

The Integrated Virtual Debugger is supported on any Workstation 7.0 virtual machine that is running a supported Windows or Linux guest operating system.

Supported Guest Operating Systems

This section provides a simplified list of guest operating systems supported for debugging in virtual machines. For guest operating system support, known issues, and installation instructions, see the online *VMware Compatibility Guide*. Go to the VMware Web site and select **Resources > Compatibility Guides**, and click the **View the Guest/Host OS tab on the VMware Compatibility Guide Web site** link. This guide also provides notes on installing the most common guest operating systems.

NOTE   If an operating system is not listed in this table, it does not support debugging in a virtual machine.

The Integrated Virtual Debugger supports the following Windows 32-bit, Windows 64-bit, Linux 32-bit, and Linux 64-bit guest operating systems.

Table 2-1. Windows and Linux Guest Operating Systems

Processor Type	Operating System Edition
Windows 32-bit	Windows 7 Ultimate Edition
	Windows 7 Enterprise Edition
	Windows 7 Professional
	Windows 7 Home Premium
	Windows 7 Home Basic
	Windows Vista (all editions except Vista Home Edition, which cannot be run in a virtual machine because of Microsoft licensing restrictions.)
	Windows Server 2003
	Enterprise Edition and R2
	Windows XP Professional
	Windows Home Edition
	Windows 2000 Professional
	Windows 2000 Server
	Windows 2000 Advanced Server

Table 2-1. Windows and Linux Guest Operating Systems (Continued)

Processor Type	Operating System Edition
Windows 64-bit	Windows 7 Ultimate Edition
	Windows 7 Enterprise Edition
	Windows 7 Professional
	Windows 7 Home Premium
	Windows 7 Home Basic
	Windows Vista x64 Edition (Aero and 3-D effects not yet supported)
	Windows Server 2003 x64 Edition
	Windows XP Professional x64
Linux 32 and 64-bit	Red Hat Linux 9
	Red Hat Enterprise Linux 5.x
	Red Hat Enterprise Linux 4.x
	Red Hat Enterprise Linux Advanced Server
	Enterprise Server
	Workstation 4 and 5
	Ubuntu Linux 6.xx, 8.04, 8.10, and 9.04
	SUSE Linux 10
	SUSE Linux Enterprise Server 10

Update the Eclipse Build Settings to Use a 1.4.x JRE

You cannot have GCJ installed on the guest operating system.

The guest operating system must be running JRE 1.4.2 or higher. If you are not using JRE 5.0 on the guest, you must update the build settings in Eclipse to use a 1.4.x JRE on the guest.

To update the Eclipse build settings to use a 1.4.x JRE

1   In the Eclipse Package Explorer, right-click the topmost folder (Project item) and choose **Properties**.

2   In the left pane of the Properties page, select **Java Compiler**.

3   Select **Enable project specific settings**, and set the Java Development Kit (JDK) Compliance Compiler compliance level to 1.4.

Installing PSAPI.DLL on Windows NT

On Windows NT, you must install the psapi.dll library file to retrieve process status information so that the Integrated Virtual Debugger can attach to a process. You can download psapi.dll from the Microsoft Developer Network.

Disabling the Firewall on Linux Guest Systems

You must disable the firewall on Linux guest operating systems. The Integrated Virtual Debugger opens an available port (searching from port 49152) for each debugging session.

Configure the Firewall on Windows Guest Systems

If you are using a 1.4.x JRE on Windows guest systems, you must disable the firewall or allow incoming connections to the JVM. If your Windows system (such as Windows XP SP2, Windows 2003, and Windows Vista) allows you to configure exceptions to the firewall, you can add the JVM to the exceptions list.

To configure the firewall on Windows guest systems

1   Choose **Start > Control Panel > Windows Firewall** and select the **Exceptions** tab.

2   Click **Add Program** and browse to the Java executable.

3   Click **OK**.

4   (Optional) On Windows Vista guests, you might have to restart the firewall after configuring it to allow
     incoming connections to the JVM.

Managing Virtual Machine Launch Configurations

You can manage configuration settings for each virtual machine in which you want to debug applications.
Integrated Virtual Debugger launch configurations determine which virtual machine to run the application in
and how the application is executed.

The launch configuration types **VMware attach to application** and **VMware execute Java application** have a
VMware tab. The values you enter in the **VMware** tab determine virtual machine configuration settings. After
you configure a virtual machine, you can start applications in virtual machines, and attach to them, from the
Eclipse **Debug** and **Run** menus.

Use Application Configurations to Start Applications in a Virtual Machine

This section describes how to create, duplicate, or edit a launch configuration to start an application in a virtual
machine.

To use application configurations to start applications in a virtual machine

1   Choose **Run > Debug.**

     The **Debug** page is displayed. You can create, manage, and run configurations from this page.

2   To create a launch configuration or edit an existing configuration, do one of the following:

     ■   Create a configuration based on default settings by selecting **VMware execute Java application** in the
         left pane, and clicking the **New launch configuration** icon at the top of the pane.

     ■   Create a configuration based on another configuration by selecting the configuration you want to
         duplicate under **VMware execute Java application** in the left pane, and clicking the **Duplicates the
         currently selected configuration** icon at the top of the pane.

     ■   Edit an existing configuration by selecting the configuration you want to edit under **VMware execute
         Java application** in the left pane.

3   Do the remaining steps in the **VMware** tab of the right pane.

4   Browse to choose a virtual machine from the drop-down menu of recently used and currently running
     virtual machines.

5   Enter your account credentials to access the guest console.

6   (Optional) If you want to use a JVM other than the one that is automatically selected, select an alternate
     JVM path.

7   (Optional) Expand the list of shared folders to add, edit, or remove folders to be shared between the host
     and the guest systems.

     For each folder, enter the share name and the location on the host system. By default, the project folder is
     shared.

8   (Optional) Indicate actions to be performed before the application is launched:

     ■   Select **Set virtual machine state to most recent snapshot** to revert to the most recent snapshot before
         the application is launched.

     ■   Select **Run script** and enter one or more shell commands to be executed in the guest operating system
         before the application is launched. No syntax checking is performed. Either enter one command per
         line, or enter multiple commands on the same line using a semicolon as a separator.

9   (Optional) Indicate actions to be performed after the application has terminated:

- Select **Run script** and enter one or more shell commands to be executed in the guest operating system after the application has terminated. No syntax checking is performed. Either enter one command per line, or enter multiple commands on the same line using a semicolon as a separator.

- Select one of the **Set virtual machine state** options.

10   Click **Apply**.

If newly created, the launch configuration is added to the left pane.

Use Configurations to Attach to Applications Running in a Virtual Machine

This section describes how to create, duplicate, or edit a configuration that attaches to a running application in a virtual machine.

To use application configurations to attach to applications running in a virtual machine

1   Choose **Run > Debug**.

2   Do one of the following tasks:

- Create a configuration based on default settings by selecting **VMware attach to application** in the left pane, and clicking the **New launch configuration** icon at the top of the pane.

- Create a configuration based on another configuration by selecting the configuration you want to duplicate under **VMware attach to application** in the left pane and clicking the **Duplicates the currently selected configuration** icon at the top of the pane.

- Edit an existing configuration by selecting the configuration you want to edit under **VMware attach to application** in the left pane.

3   Do the remaining steps in the **VMware** tab of the right pane.

4   Browse or choose a virtual machine from the drop-down menu of recently used and currently running virtual machines.

5   Enter your account credentials to access the guest console.

6   Click **Apply**.

After you create a new launch configuration, it appears in the left pane.

Delete a Configuration

Before you delete a configuration, make sure the virtual machine is powered off or suspended.

To remove a configuration

1   Choose **Run > Debug**.

2   In the left pane, select the configurations that you want to delete and click the **Delete selected launch configuration(s)** icon at the top of the pane.

Running and Debugging Applications in Virtual Machines

You can use the Integrated Virtual Debugger to:

- 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.

Start an Application Debugging Session in a Virtual Machine

Do not suspend a virtual machine while the Integrated Virtual Debugger is connected to an application. If you do, the Integrated Virtual Debugger disconnects from the application.

To start an application debugging session in a virtual machine

1   Begin the session in one of the following ways:

   ■   From the **Debug** menu, choose the configuration for the application to start debugging.

   ■   In the **Debug** page, select the configuration under **VMware execute Java application** in the left pane and click **Debug** in the right pane.

2   Perform debugging tasks as you would in a local debugging environment.

Start an Application in a Virtual Machine Without Debugging

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

To start an application in a virtual machine without debugging

Begin the session in one of the following ways:

■   From the **Run** menu, choose the configuration for the application to start.

■   In the **Run** page, select the configuration under **VMware execute Java application** in the left pane and click **Run** in the right pane.

Attach to an Application Running in a Virtual Machine

Do not suspend a virtual machine while the Integrated Virtual Debugger is connected to an application. If you do, the Integrated Virtual Debugger will disconnect from the application.

To attach to an application running in a virtual machine

1   In the **Debug** page, select the configuration under **VMware attach to application** in the left pane and click **Debug** in the right pane.

2   Select the process you want to attach to.

   If more than one instance of the Java application is running in the virtual machine, a dialog box appears with a list of the running instances, each identified by their process ID, port number, and arguments.

3   Perform debugging tasks as you would in a local debugging environment.

Remote Debugging in the C and C++ Environment

3

You can use remote debugging to debug C and C++ applications from inside a virtual machine. You can set a remote debugging session inside a controlled C and C++ environment where the guest operating system is different from the host. This chapter includes the following sections:

- "Virtual Machine Requirements for C and C++ Applications" on page 17
- "Managing Virtual Machine Launch Configurations" on page 20
- "Running and Debugging Applications in Virtual Machines" on page 21
- "Troubleshooting the C and C++ Environment" on page 21

Virtual Machine Requirements for C and C++ Applications

The Integrated Virtual Debugger is supported on any Workstation 7.0 virtual machine that is running a supported Windows or Linux guest operating system.

Supported Guest Operating Systems

The Integrated Virtual Debugger supports the following Windows 32-bit, Windows 64-bit, Linux 32-bit, and Linux 64-bit guest operating systems. Table 3-1 provides a simplified list of guest operating systems for which remote debugging in virtual machines is supported.

For guest operating system support, known issues, and installation instructions, see the online *VMware Compatibility Guide*. Go to the VMware Web site and select **Resources > Compatibility Guides**, and click the **View the Guest/Host OS tab on the VMware Compatibility Guide Web site** link. This guide also provides notes on installing the most common guest operating systems.

NOTE If an operating system is not listed in this table, it does not support debugging in a virtual machine.

Table 3-1. Windows and Linux Guest Operating Systems

Processor Type	Operating System Edition
Windows 32-bit	Windows 7
	Windows Vista (all editions except Vista Home Edition, which cannot be run in a virtual machine because of Microsoft licensing restrictions.)
	Windows Server 2008
	Windows Server 2003 Enterprise Edition and R2
	Windows XP Professional Windows Home Edition

Table 3-1. Windows and Linux Guest Operating Systems (Continued)

Processor Type	Operating System Edition
Windows 64-bit	Windows 7 x64 Edition
	Windows Vista x64 Edition
	Windows Server 2008
	Windows Server 2003 x64 Edition
	Windows XP Professional x64
Linux 32 and 64-bit	Red Hat Linux 8 and 9
	Red Hat Enterprise Linux 5.x Red Hat Enterprise Linux 4.x Red Hat Enterprise Linux Advanced Server Enterprise Server Workstation 4 and 5
	Ubuntu Linux 8.04, 8.10, and 9.04
	Open SuSE Linux 10.x SuSE Linux Enterprise Server 11.x

Install Eclipse Classic 3.4.x on the Host

Eclipse Classic 3.4.x and higher includes the Eclipse Platform, Java Development Tools and plug-in development environment.

Before you begin, download Java 6 Update 12 or higher.

To install Eclipse Classic 3.4.x on the host

1   Download the latest version of the Eclipse Classic 3.4.x.

The latest version is available for download on the eclipse.org Web site.

2   Unzip the downloaded file and open the folder.

3   Select the `eclipse.exe` file and follow the prompts.

4   Browse and select a directory for your workspace folder.

Use Proxy for Network Connection

If you are using a proxy for network connection, you must set the network proxy.

To use proxy for network connection

1   Select **Window > Preferences.**

2   Type **Network Connections** in the search box.

3   Select the **Network Connection** menu that appears in the right-hand panel.

4   Select **Manual proxy configuration**.

5   Enter the **HTTP proxy** address and **Port** number.

6   Check the **Use this proxy server for SSL** box.

7   Select **Apply** and click **OK**.

Install C and C++ Development Toolkit and Remote System Explorer Plug-ins

The Eclipse Classic 3.4.x requires the C and C++ development toolkit (CDT) and remote system explorer (RSE) plug-ins to debug C and C++ applications.

To install C and C++ development tooling and remote system explorer plug-in

1   Select **Help > Software Updates**.

2   Click on the **Available Software** tab.

3   From the list of available software, check **Ganymede** and click the expand button.

4   Under **Ganymede** select the software for installation.

    a   **C and C++ Development > Eclipse C/C++ Development Platform**

    b   **C and C++ Development > Eclipse C/C++ Development Tools**

    c   **Remote Access and Device Development > Remote System Explorer C/C++ Remote Debug Launcher**

    d   **Remote Access and Device Development > Remote System Explorer End-User Runtime**

5   Select **Install** and click **Finish**.

Install the Eclipse Plug-in Manually

You must install the Eclipse plug-in manually if you did not select to install the Eclipse plug-in when you installed VMware Workstation, or if your Eclipse application was not configured properly and errors during installation of Workstation.

To install the Eclipse plug-in manually

1   Start the latest version of the Eclipse Classic 3.4.

2   Select a workspace.

3   Select **Help > Software Updates**.

4   Select the **Available Software** tab and click the **Add Site** button on the right side.

5   Select **Archive** to browse to the location of the plug-in `zip` file and click **OK**.

- On Windows, the file is located in
  `C:\Program Files\eclipse\VMwareEclipseCCPP\vmware-eclipse-update-site.zip`

- On Linux, the file is located in
  `/usr/lib/vmware/eclipse-ng/vmware-eclipse-update-site.zip`

6   Under the **Name** section, check the **VMware C/C++ Virtual Machine Remote Debugging** option**.**

7   Click the **Install** button on the right side**.**

8   Click **Finish** and select **Yes** to restart the Eclipse application.

You can now open the Debug configuration dialog and create a VMware C/C++ configuration. See "Managing Virtual Machine Launch Configurations" on page 20.

Disabling the Firewall on Guest Systems

You must disable the firewall on Windows and Linux guest operating systems. The Integrated Virtual Debugger opens an available port (searching from port 49152) for each debugging session.

Managing Virtual Machine Launch Configurations

You can manage the Debug configuration settings to determine which virtual machine runs the application and how the application is run. The configuration also allows you to run command-line scripts before and after the debugging session and enable specified shared folders during the debugging session. After you configure the debugger, you can start applications in virtual machines from the Eclipse **Debug** and **Run** menus.

Set Debug Configuration Settings to Start Applications in a Virtual Machine

You can create, manage, or run a launch configuration to start an application in a virtual machine. Before you begin, make sure the virtual machine is powered on.

To set debug configuration settings to start applications in a virtual machine

1　Choose **Run > Debug Configuration**.

2　In the Virtual Machine Options section, select the **Main** tab to create or edit the **Virtual Machine Path**, **Gdbserver Path,** or **Start From Snapshot** text box**.**

- To set a **Virtual Machine Path**, click **Browse** and select a recently used virtual machines.

- To set a **Gdbserver path**, click **Browse** and select path to the gdbserver executable file.

- (Optional) To revert to the most recent snapshot before the application is launched, click the **Start From Snapshot** check box and enter your login credentials in the Authentication dialog box.

    Click **Refresh** to refresh the current snapshot list.

3　(Optional) In the Virtual Machine Options section, select the **Shared Folders** tab to add, edit, or remove shared folders between the host and the guest systems. For each folder, enter the folder name and the host path location on the host system. By default, the project folder is shared.

4　(Optional) In the Virtual Machine Options section, select the **Scripts** tab to run script before launching and after terminating the application.

- In the **Before Launch, Run script** text box, enter one or more shell commands to run in the guest operating system before the application is launched.

- Select **After Termination, Run script** text box, enter one or more shell commands to run in the guest operating system after the application has terminated.

    No syntax checking is performed. Either enter one command per line, or enter multiple commands on the same line using a semicolon as a separator.

5　To make the Eclipse debugging user interface available for debugging, select the **Debugger** tab and set the GDB debugger path.

- In the **Main** tab, to complete the GDB debugger text box click Browse and select path to the gdb executable file.

- In the **Main** tab, accept the default value in the GDB command file text box.

6　Click **Apply**.

After you create a launch configuration, it appears in the left pane.

Delete a Configuration

If you do not need a configuration or want to clear memory space, you can delete the configuration.

To remove a configuration

1　Choose **Run > Debug Configuration**.

2　In the left pane, select the configurations that you want to delete and click the **Delete selected launch configuration(s)** icon at the top of the pane.

Running and Debugging Applications in Virtual Machines

You can use the Integrated Virtual Debugger to:

- Start an application debugging session in a virtual machine.

- Start an application in a virtual machine without debugging.

Start an Application Debugging Session in a Virtual Machine

Do not suspend a virtual machine while the Integrated Virtual Debugger is connected to an application. If you do, the Integrated Virtual Debugger disconnects from the application.

To start an application debugging session in a virtual machine

1   You can start debugging in the following ways:

- Click **Run > Debug Configurations** and under **C/C++ Attach to Virtual Machine**, choose the configuration.

- Click the **Debug** button and choose the configuration either from the drop-down menu (if previously debugged) or click the **Debug** button and click **Debug configuration**.

2   Perform debugging tasks as you would in a local debugging environment.

Start an Application in a Virtual Machine Without Debugging

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

To start an application in a virtual machine without debugging

Begin the session in one of the following ways:

- Click the drop-down arrow on the **Run** button and choose the configuration for the application that you want to start. (The **Run** menu does not provide a way to choose the configuration.)

- In the **Run** page, select the configuration under **C/C++ Attach to Virtual Machine** in the left pane and click **Run** in the right pane.

Troubleshooting the C and C++ Environment

Before you troubleshoot an error make sure that you have correctly installed and configured the Integrated Virtual Debugger. You must power on a virtual machine before you can debug it.

Debugging Session Fails to Start

Problem

A debugging session fails to start.

Causes

This problem has several causes:

- The firewall is turned on in the guest.

- The network adapter is disabled, or networking functionality is not ready in the guest.

- The virtual machine is powered off.

- The Ubuntu 8 host is causing the debugging session to fail.

Solutions

The following actions provide solutions for each of the causes:

- Turn off the firewall in the guest.

- Make sure that networking is enabled and functional in the guest.

- Power on the virtual machine and start a debugging session.

- Run the following command:
  `./eclipse –vmargs –Dorg.eclipse.swt.internal.gtk.disablePrinting`

Debugging Session Fails to Start and Protocol Error Appears

Problem

A debugging session fails to start, and the error message **Protocol Error** or the error message `can't execvp /mnt/hgfs/gdbserverDirectory/gdbserver` appears. The error appears on any Linux 64-bit host with any 32-bit Linux guest.

Solution

To work around this issue, switch to any 64-bit Linux guest and start a debugging session.

Eclipse Fails When You Start a Debugging Session

Problem

The Eclipse application fails when you start a debugging session.

Cause

The cause might be that the Eclipse and SWT (GUI toolkit) are incompatible.

Solution

To work around this issue, run the following Eclipse command:

`./eclipse –vmargs –Dorg.eclipse.swt.internal.gtk.disablePrinting`

Reverting to the Recent Snapshot Stops Responding

Problem

The Eclipse application stops responding when the **Start from Snapshot** check box is selected.

Cause

The Eclipse application is waiting until the virtual machine is powered on.

Solution

Make sure that the virtual machine is powered on before you select the **Start from Snapshot** option.

Log in Authentication Error on the Guest

Problem

When you provide your authentication information to log in to the guest, you receive an error message that tells you to log in interactively from the guest.

Causes

The following issues are the main causes:

- The Java or C and C++ plug-in was unable to log in.

- The account used has no password protection.

Solutions

To work around this issue, do one of the following tasks:

■ Log in using the guest operating system login screen before you continue the debugging session.

■ You must use an account that is password protected.

Index