



Microsoft Java Virtual Machine Transition Guide for IT Professionals

Version 2.3

July 24, 2004

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This guide is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2004 Microsoft Corporation. All rights reserved.

Microsoft, Windows, Windows Server, Visual J#, Visual J++, Visual J#, Visual C#, ActiveX, JScript, Visual Basic and Visual Studio are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Contents

| | | |
|---|-------|----|
| 1 | | 5 |
| Overview | | 5 |
| Intended Audience | | 5 |
| How to Use This Solution Guide | | 6 |
| Why a Transition Is Needed | | 7 |
| Are You Affected? | | 7 |
| IT Professionals Take Action | | 8 |
| Assessing Your Situation | | 8 |
| Security Considerations after December 31, 2007 | | 9 |
| Microsoft Product Support Services | | 9 |
| Example Process | | 9 |
| 2 | | 11 |
| The Diagnostic Tool for the Microsoft VM | | 11 |
| Types of Scan Modes | | 12 |
| Scan Options | | 12 |
| Diagnostic Results | | 13 |
| Report Statistics | | 13 |
| Scan Options | | 13 |
| File Scan Summary | | 14 |
| URL Scan Report | | 14 |
| File Scan Details | | 14 |
| Interpreting Diagnostic Results | | 15 |
| Example Methodology for Processing Result Data | | 15 |
| 3 | | 17 |
| Available Transition Options | | 17 |
| Remove Applets | | 17 |
| Lock Down Security without Altering Applications | | 18 |
| Migrate or Rewrite Applications | | 18 |
| Migrate to .NET and Visual J# (J# Browser Controls) | | 19 |
| Migrate to .NET and Visual C# (JLCA) | | 19 |
| Use an Alternate Rendering Technology | | 19 |
| Use an Alternate JRE | | 20 |
| 4 | | 21 |
| Lock Down Security | | 21 |
| Locking Down Internet Explorer | | 21 |
| Pointing to the MSJVM | | 21 |
| 5 | | 23 |
| The MSJVM Removal Tool | | 23 |
| 6 | | 25 |
| Switch to Another JRE | | 25 |

1

Overview

In a settlement agreement reached in January 2001 to resolve a dispute over Microsoft's distribution of its Java implementation, Sun and Microsoft agreed to limit the duration of Microsoft's use of Sun's source code and compatibility test suites to support the Microsoft® Java Virtual Machine (MSJVM). However, as part of the April 2004 settlement, Sun and Microsoft agreed to extend Microsoft's license to use Sun's Java source code and compatibility test suites. This extension allows Microsoft to support the MSJVM and address potential security issues until December 31, 2007.

Consistent with the agreements made with Sun, Microsoft has been phasing out the MSJVM in its products. Going forward, the MSJVM will not be included in any future Microsoft products. The MSJVM is obsolete code and will not be enhanced or developed. Microsoft will only continue to provide security fixes to help keep customer machines secure.

Sites and applications that currently depend upon the MSJVM will not function correctly when accessed by systems that do not have the MSJVM installed. Users with updated Microsoft products without the MSJVM may experience problems with Internet or Intranet sites requiring the MSJVM.

Intended Audience

This is a technical guide intended for IT professionals responsible for identifying sites and applications that require the MSJVM or contain Java-language code, and planning a transition to alternate solutions.

IT professionals will find this guide useful in identifying dependencies on the MSJVM. IT business decision makers or those charged with establishing a technology strategy will find useful information for defining a strategy to transition away from any dependencies. Technical decision makers will find guidance on the implementation of that strategy.

The *Microsoft Java Virtual Machine Migration Guide for Developers* is available for those tasked with defining the types of changes that need to be made, determining the appropriate approach to the work, and implementing these changes.

How to Use This Solution Guide

This document provides specific guidance for MSJVM transition projects. The guide begins with the information you need to assess your situation, a description of your planning options, and details about each option.

The following is a list of content for this guide's chapters:

Chapter 1—Overview. This chapter outlines the guide and describes why support is being discontinued for the MSJVM, how this may impact users and businesses, and a recommended plan of action for IT professionals.

Chapter 2—The Diagnostic Tool for the Microsoft VM. This chapter describes the tool available for automatically assessing all dependencies to the MSJVM, how to interpret diagnostic reports, and how to process the resulting data.

Chapter 3—Available Transition Options. This chapter describes possible transition options and outlines factors in planning a transition from the MSJVM.

Chapter 4—Lock Down Security. This chapter describes a methodology for maintaining use of the MSJVM on legacy systems.

Chapter 5—The MSJVM Removal Tool. This chapter describes the tool available for removing the MSJVM.

Chapter 6—Switch to Another JRE. This chapter discusses general topics involved in selecting, installing, and deploying an alternate JRE.

Why a Transition Is Needed

The future of the MSJVM is clearly defined by the January 2001 settlement with Sun Microsystems, which contemplates a phase out of the MSJVM. Microsoft and Sun have entered into an April 2004 settlement, which permits Microsoft to continue support of the MSJVM until December 31, 2007. After December 31, 2007, Microsoft will be unable to update the MSJVM, even to repair critical security vulnerabilities and customer defects. Although Microsoft will continue to support customers, the ability to do so will be extremely limited. In light of these restrictions, new Microsoft products will not include the MSJVM.

After the deadline, the MSJVM could be vulnerable to security issues that will be impossible for Microsoft to remediate. Businesses depending on the MSJVM that failed to migrate to alternate solutions may be in a position wherein they are required to disable the MSJVM completely with limited notice. As a result, mission-critical applets and J++ applications may cease to function properly, potentially impacting business operations and customers.

Are You Affected?

Many IT Professionals are unaware of dependencies they may have on the MSJVM, of the deadline for end of support, and the potential security risks that may occur once Microsoft loses the ability to support the MSJVM. Microsoft stresses the importance of each customer making an effort to understand the current situation with the MSJVM and the impact it may have on their day-to-day activities.

A basic assessment to understand whether this change affects you can be as simple as answering these questions:

Do you use any Microsoft applications with known MSJVM dependencies?

Do you use any third-party applications with known MSJVM dependencies?

Have you created any custom applications with MSJVM dependencies?

If you answer “No” to all three of the above, then your solution is to remove the MSJVM from your environment using the MSJVM Removal Tool. However, if you answer “Yes” or are unsure of an answer to any of these questions, it is necessary to initiate a discovery process for MSJVM dependencies in your environment.

Microsoft applications that were identified as having dependencies on the MSJVM will either be retired or re-released without the dependency. If you use a third party application with MSJVM dependencies, contact the vendor for information on how they plan to address the issue.

To review the list of known Microsoft and third-party applications with MSJVM dependencies refer to: <http://www.microsoft.com/java>.

Note The lists of applications provided at the link above are intended to provide you with a starting point for MSJVM dependency discovery. If a particular application is not listed, it does not necessarily mean that it does not have a MSJVM dependency. Evaluations on MSJVM dependencies are ongoing.

IT Professionals Take Action

Because you may have dependencies in a variety of websites and applications, Microsoft strongly recommends that IT professionals determine the extent of their dependencies on the MSJVM. Keep in mind that any solution chosen to address discovered dependencies may require extensive testing and an understanding of the range of systems affected.

It is important to identify all applets and Java-dependent applications and components:

- Applets are identified by the <APPLET>, <OBJECT>, or <EMBED> tags, either directly or through DLLs or ActiveX controls.
- Java-dependent applications and application components are applications that make use of the MSJVM, either directly or through DLLs or ActiveX controls, including the J++ source files and the binary files. Possible binary file extensions include .exe, .ocx, and .dll.

Some technologies may appear as though they have MSJVM dependencies, but this is not the case. For example the following are not affected, except when they use one of the affected technologies mentioned above:

- JavaScript, Microsoft JScript®, Visual J#, or ECMAScript usage on any site.
- Websites that use server-side Java technologies, such as Java Server Pages (JSP) or Servlets.

Assessing Your Situation

The Diagnostic Tool for the Microsoft VM scans systems and produces a report on the resources that have dependencies on the MSJVM or Java-language code. By analyzing the diagnostic report you can identify if the MSJVM is installed, what types of dependencies exist, and what applications and websites are affected.

Once you identify dependencies on the MSJVM, Microsoft recommends that you take one or more of the following actions, as appropriate:

- Remove the MSJVM.
- Lock down security and restrict access to applets or applications.
- Migrate to another solution as discussed in the *Microsoft Java Virtual Machine Migration Guide for Developers*:
 - Visual J# .NET or J# Browser Controls.
 - C# on .NET, using the Java Language Conversion Assistant.
 - Other rendering technologies, such as DHTML or a 3rd party display technology such as Flash or Shockwave.
 - An alternate JRE.

Note Results of the discovery process may indicate that you have Java dependencies, but do not have MSJVM dependencies. It is important to understand this distinction. Organizations with Java dependent applications and applets that do not use MSJVM-specific features may want to consider switching to an alternate JRE.

Security Considerations after December 31, 2007

Although a possible option for organizations after December 31, 2007 is to do nothing to resolve MSJVM dependencies, this is not recommended by Microsoft. The MSJVM will become unsupported software on December 31, 2007, and as a result, there are serious security implications of not transitioning MSJVM dependencies before this date.

Running the MSJVM as unsupported software exposes your system to the possibility of security vulnerabilities. If a security issue is identified following the deadline, Microsoft will be unable to make security or functionality patches to the MSJVM available to users.

Prior to the deadline, every effort should be made to remove or transition away from the MSJVM. At a minimum, if you are unable to successfully transition or decide to not transition away from the MSJVM, locking down security for the MSJVM to trusted sites is recommended.

The MSJVM Removal Tool is available to remove the MSJVM from client systems after transitioning. Removing the MSJVM is one way to help ensure you will not be vulnerable to any security risks related to the MSJVM.

Microsoft Product Support Services

Microsoft is committed to helping customers through this transition and offers migration support at no charge for the Diagnostic Tool for the Microsoft VM, the MSJVM Removal Tool, and the Java Language Conversion Assistant.

Additionally, Microsoft Product Support Services (PSS) is available to help assist migrating existing applications and functionality to any Microsoft technology, including the .NET environment. This enables developers to migrate existing applications with current feature sets. Technical assistance on adding additional functionality is not included and will continue to be fee-based. MSJVM migration support is now available worldwide. Please contact PSS at **800-936-5800** in the US or your local subsidiary.

Note Microsoft will continue support of existing products that have MSJVM dependencies according to existing support policies, but cannot support MSJVM related issues after December 31, 2007. For more information on International Support, refer to: <http://support.microsoft.com/common/international.aspx>.

Example Process

The following graphic (Figure 1.1) is an example of how to initiate the process of dependency discovery on the MSJVM and what options are available to address those dependencies. Each step in the example process is described in more detail throughout this manual.

Note This graphic is intended to provide a starting point from which you can plan your own assessment and solutions based on your business needs and environment, and does not include steps such as re-running the Diagnostic Tool, testing migration options, etc.

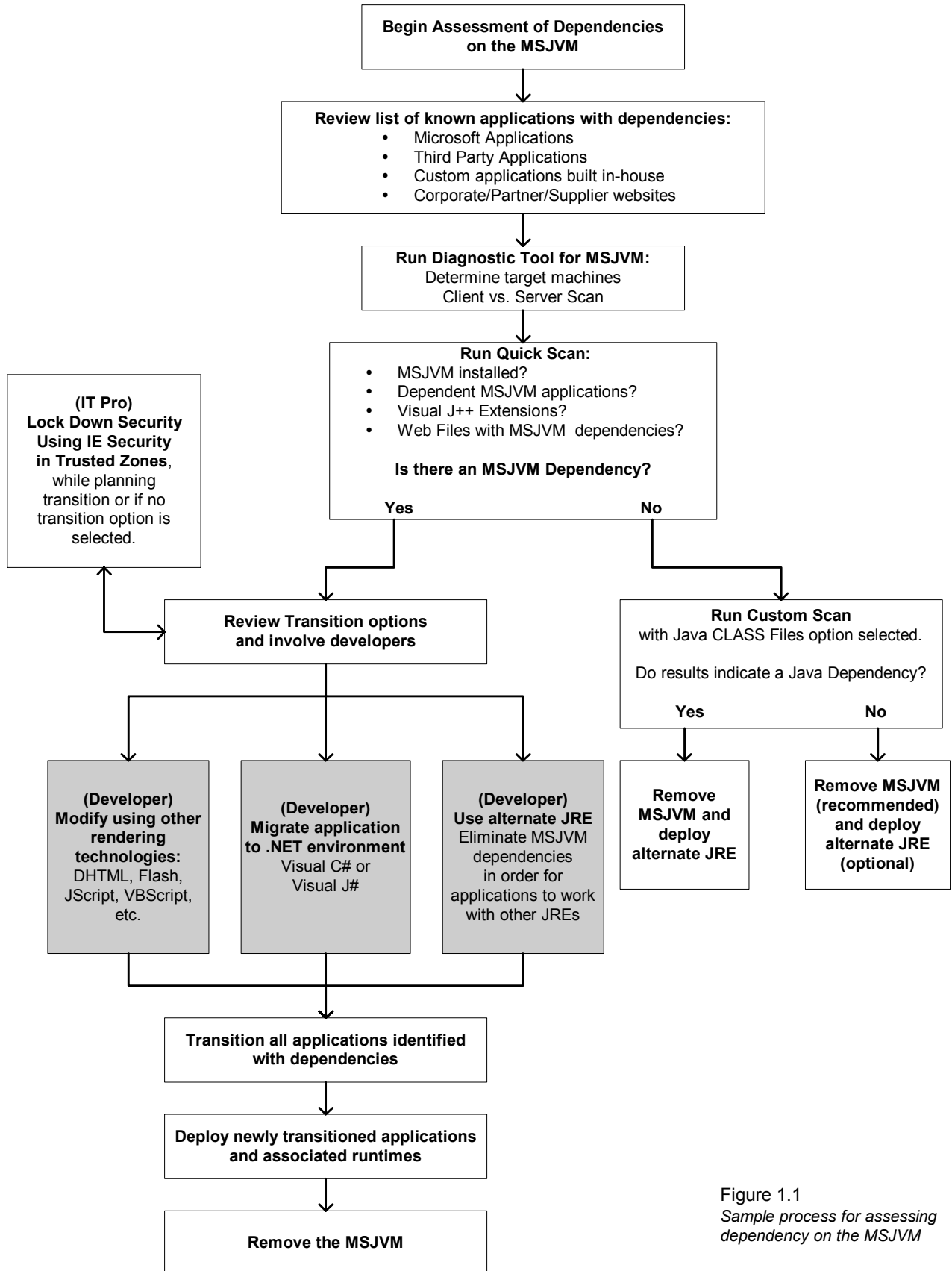


Figure 1.1
Sample process for assessing dependency on the MSJVM

2

The Diagnostic Tool for the Microsoft VM

Note For complete instructions on installation, use, and the system requirements of this tool, refer to the *Diagnostic Tool for the Microsoft VM User Manual*. The MSJVM website contains information about obtaining this tool: <http://www.microsoft.com/java>.

The Diagnostic Tool for the Microsoft VM searches for evidence of MSJVM dependencies. It scans targeted machines for the following types of dependencies:

- Existence of the MSJVM, including registry entries.
- MSJVM-dependent applications (including files with the .exe, .dll, and .ocx extensions).
- Compiled Java files (with the .class extension).
- Web page files (with extensions .html, .jsp, .php, .asp, and so on) that contain or invoke the MSJVM.

This tool is intended to simplify the process of locating all dependent applications and web pages by scanning machines and compiling the results in a single report. Ideally, the Diagnostic Tool will be run several times during the course of discovery and transition: first, to assess all dependencies prior to attempting a transition; second, during the transition project as a guide and reminder of current status; and third, during deployment to ensure all applets and applications were identified and transitioned.

To run an effective scan, make sure that you are focusing the scope of the scan on the highest priority issues. For example, you may want to first identify all MSJVM-specific dependencies using the Quick Scan, and at a later time run a separate scan for all Java dependencies, as well as scan URLs for Java dependencies in web pages using a Custom Scan.

The Diagnostic Tool can create a large amount of data. One of the best ways to reduce resulting data is to scan as few machines as possible to get the necessary results. It is rarely necessary to scan every machine in an enterprise. If your environment contains one or more standard desktops that cannot be modified by the end user, then scanning a single copy of the installed images will identify dependencies present on all desktops.

Additionally, if there are particular groups of machines that are similar in setup to each other—such as all of your web servers—scanning representative machines may be an option. By selecting representative samples of machines from your enterprise and limiting the number of scans run, you can greatly reduce the amount of extraneous data and avoid being overwhelmed by redundancies in the diagnostic reports.

As you progress and resolve dependencies, the number of issues found during each subsequent scan will decrease. At that point you can choose to expand the number of machines in your sample size to check for additional applications that are in need of remediation.

Types of Scan Modes

The Diagnostic Tool prompts you to specify controls of two types: the first controls which computers are examined, and the second controls the types of dependency that are searched for. Any set of computers on the local network can be specified, but you will need administrator-level privileges to run the tool locally and remotely. There are three types of scan modes initiated by the Diagnostic Tool:

Quick Scan

The Quick Scan is ideal for identifying your MSJVM dependencies. It scans for the MSJVM and Microsoft SDK for Java; MSJVM dependent applications; and web files containing applets or MSJVM references. Quick Scan includes only the following scan options: File Scan – Runtime, File Scan – Dependent Microsoft VM Applications, File Scan—Compiled Visual J++ Extensions and File Scan – Web Files with Microsoft VM References.

Complete Scan

A Complete Scan scans for the MSJVM and Microsoft SDK for Java; MSJVM dependent applications; java CLASS files; compiled java code using the Microsoft language extensions for java; web files containing applets or MSJVM references; and URLs containing applets or MSJVM references. This scan identifies MSJVM and Java dependencies. The Complete Scan includes all scan options listed below.

Custom Scan

A Custom Scan allows you to choose which scan options you would like to perform from the scan options listed below.

Scan Options

- **File Scan—Runtime**
This option scans the registry for the presence of the MSJVM and the Microsoft SDK for Java. If the MSJVM is installed, the version and filename of the VM DLL is displayed in the report. This option is available in all three scan modes.
- **File Scan—Dependent Microsoft VM Applications**
This option scans the logical drives of all target machines for compiled, executable applications that require the MSJVM to run, including EXE, DLL, and OCX. Files are searched for embedded tokens that indicate that the application may be linked to the MSJVM. This option is available in all three scan modes.
- **File Scan—Compiled Visual J++ Extensions**
This option scans for all compiled applications that contain MSJVM specific dependencies, including EXE, DLL, OCX, and CLASS files. These MSJVM features are referred to as Visual J++ Extensions. These include RNI, Java/COM

Interop, and JDirect. It also finds Java classes registered as COM components. This option is available in all three scan modes.

- **File Scan—Web Files with Microsoft VM References**

This option scans the target machine for all web files that contain Java applets that may depend on the MSJVM. The files are searched for <APPLET>, <EMBED>, and <OBJECT> tags that result in a Java applet appearing in the web browser. This lists all applets, not only those with MSJVM dependencies. It is possible that an alternate JRE would also run these applets. This option only checks files on the selected target computers. This option is available in all three scan modes.
- **File Scan—Java CLASS Files**

This option scans for Java bytecode CLASS files and JAR (Java archive) files. This means that all non-Microsoft Java CLASS files are also detected. Files with MSJVM requirements are detected in the Compiled Visual J++ Extensions scan. Files listed as a result of this scan, but were not reported in the Compiled Visual J++ Extensions scan are candidates for running on an alternate JRE. This option is only available in Complete Scan and Custom Scan.
- **URL Scan**

This option scans a URL for all web files that contain Java applets that may depend on the MSJVM. This scan works essentially the same as the Web File scan, except it downloads the pages from the web using the given URLs. This scan works like a web crawler where the tool follows all links within the given domain to a specified depth. The default depth is “0” where only the given URL is scanned. The maximum scan depth is unlimited where all links are followed until every page within the given domain that can be accessed via the provided URL is scanned. This option is only available in Complete Scan and Custom Scan.

Note A Custom Scan with **all** scan options selected is the same as a Complete Scan.

Diagnostic Results

An automatically generated HTML report containing diagnostic results is produced by the Diagnostic Tool for the Microsoft VM. The results are summarized in the following manner: Report Statistics, Scan Options, File Scan Summary, URL Scan Report, and an individual report for each scan option chosen on each target machine.

Report Statistics

The Report Statistics section includes general information such as when the results of the report were generated, the name of the target machine(s), the start and stop time of the scan, the duration of the scan, and whether or not any errors occurred during the course of the scan.

Scan Options

The Scan Options section specifies what scan options were run by the Diagnostic Tool:

VM Runtime—Specified as “Yes” if this scan option was selected.

Microsoft VM Dependency—Specified as “Yes” if this scan option was selected.

Java Binaries—Specified as “Yes” if this scan option was selected.

MS Java Binaries—Specified as “Yes” if this scan option was selected.

Web File Extensions—List of file extensions (i.e. html, htm, shtml, jsp, etc.) that were searched.

URL Scan – Crawl Depth—The depth of the URL Scan. For example, top page only, one level deep, two levels deep, or all pages within the domain.

File Scan Summary

The File Scan Summary lists the number of Java-related files located on each target machine and includes the following information:

Host

The name of the host on which the scan was run.

VMs Detected

Indicates if the MSJVM is detected.

Microsoft VM Dependencies

The number of MSJVM dependent executable applications.

Java Files

The number of compiled Java files discovered.

J++ Dependencies

The number of files which use Microsoft extensions for the java language.

J++ JavaReg Entries

The number of Java classes that are registered as COM objects.

Java Web Files

The number of web page files containing embedded Java applets.

Details regarding the file scans are found in individual reports that include general and specific information on each scan option run on the target machine.

URL Scan Report

This section describes the URLs with applets and Java dependencies. The URL entered in the wizard is highlighted, with any linked, sub-level URL files listed underneath. There are two columns in this section.

The left column is **URL**, which lists the URL searched; and the right column is **Content** and either contains the message: “No Microsoft VM dependencies found.” or provides the description of the dependency found.

Any errors that occurred during the URL Scan are listed in the **File Errors** subsection and provides the name of the URL at which the error occurred and a description of that error (i.e. “The server sent back HTTP Code 404 (Not Found)”).

File Scan Details

For each scan option selected, an individual report is created providing general information including: the duration of the scan, the location and file names of affected files, and any errors that occurred during the scan.

VM Scan Report

Indicates whether the MSJVM is installed, the version number, and the location of the file.

Microsoft VM Dependency Report

Lists MSJVM dependent executable applications (including files with EXE, DLL, and OCX extensions). An entry in this scan report section indicates that there is a MSJVM dependency. These files must be transitioned, as they will not run correctly with an alternate JRE.

Java Class Report

Lists all files with the CLASS and JAR extensions. An entry in this scan report indicates that there is some type of Java dependency. However, this may not necessarily be a MSJVM dependency.

Visual J++ Class Report

Lists all files with the CLASS, EXE, DLL, and OCX extensions that utilize the Microsoft extensions for Java. An entry in this scan report section indicates that there is a MSJVM dependency.

Visual J++ JAVAREG Report

Lists java classes that are registered as COM objects.

Web File Report—Lists any web page files (including the HTML, JSP, PHP, ASP, etc. extensions) on the machine that contain embedded Java applets. Similar to the Java Class Report, an entry in this scan report indicates that there is some type of Java dependency. However, this may not necessarily be a MSJVM dependency.

Some reports may contain additional notes that signify special types of results.

File Errors—Provides the file name in which the error occurred and a description of that error.

Partial Report Notification— You can evaluate scan results as the Diagnostic Tool is running. The report will notify you if the report you are reading is not complete with the Partial Report notification. This allows you to begin interpreting scan results before all scans are finished.

Interpreting Diagnostic Results

The first step in understanding the diagnostic report is identifying any “false positives” that are reported. “False positives” are files that are reported as having dependencies on Java, but may not necessarily have a specific dependency on the MSJVM. Files listed as a result of the Java CLASS Files scan, but were not reported in the Compiled Visual J++ Extensions scan are files that have a Java dependency, but not a MSJVM dependency.

One of the challenges in interpreting the diagnostic report is identifying which application contains the MSJVM dependent file. To determine which application contains the dependent file you can view the directory name the file is located in; make an assessment based on the properties of the file; or enter the file information in a search engine to search for the application on the web.

When interpreting diagnostic results, you may find files marked as dependent that you may not feel need remediation. Files users often ignore include temporary files, uninstall files, hotfixes, and similar files. Evaluate your level of comfort with these files and choose whether or not to remediate the applications that caused these files to be created.

Example Methodology for Processing Result Data

The Diagnostic Tool for the Microsoft VM may create a large amount of data depending on the type of scan options and target machines selected. The following example describes steps on how to process report data. Modify these steps according to your specific needs.

1. Load the diagnostic report data into a database.

Copy and paste the diagnostic report data into a database or spreadsheet, or use SQL Server's Data Transformation Services (DTS) to load the XML data files created by the scan into a database.

- If you are using DTS version 1.0, the XML data files are made available in the Temp folder until you click Finish in the wizard. Once you select Finish, all XML data for the scan is deleted. Only XML data files created by the wizard are stored in the Temp folder, and are named using the following syntax:

<unique GUID for the machine>_<ScanName> XML.

- If you are using DTS version 1.0a, the XML data files are created in the Temp folder during the scan. Once the scan is completed, the files created by the wizard or from the command line are saved and moved to the "Data" subdirectory in the default report directory. The default directory varies depending on your operating system:

Windows 2000 and above:

C:\Documents and Settings\[*User Name*]\My Documents\Diagnostic Tool for the Microsoft VM\Data.

Windows NT:

C:\WINNT\Profiles\[*User Name*]\Personal\Diagnostic Tool for the Microsoft VM\Data.

Windows 95 OSR2, 98 SE, and ME:

C:\My Documents\Diagnostic Tool for the Microsoft VM\Data

2. Normalize the data by replacing directory names with system variables.

You may want to replace certain directory names with variables. Replacing directory names makes it easier to locate files installed into different directories by the same application. Examples of these directories include the Windows directory, Program Files directory, Temp directory, and other locations shared across machines in your scan.

For example, replace the Windows directories reported by various machines (C:\Windows, E:\Windows, G:\Windows) with the wildcard "%windir%."

3. Match MSJVM dependent files to the appropriate applications.

Add a column to your data for "Application" and add the application name into each row as appropriate. There are two ways of determining the application:

- The first is to determine the application based on the directory name. For example, you may find several files under the subdirectory "%Windir%\System32." Knowing that these files likely belong to the operating system, you could set the application name as Windows for all of these files.
- For applications that cannot be identified by the directory name, the properties of the file may be able to identify the application that installed it. To view file properties, right-click on a file and select Properties. Click on the Summary tab to view any identifying information encoded into the file.

4. Determine transition options.

Once you have identified the list of applications in which there were MSJVM dependencies detected, determine the correct transition path for each one. For Microsoft applications, check the application dependency list for suggested transition options. For third-party applications, contact the appropriate software vendor to determine the suggested transition option. For applications written internally, involve your developers to determine the best transition plan.

3

Available Transition Options

This chapter provides a brief summary of the possible solutions available for transitioning from the MSJVM. Developers will need to refer to the *Microsoft Java Virtual Machine Migration Guide for Developers* for more in-depth information regarding migration to .NET, other rendering technologies, and alternate JREs.

In the early stages, it is a good idea to set up a pilot project that provides a proof of technology. This project should contain samples of each type of applet to be converted, and should be restricted in scope.

From an organizational point of view, there are great benefits to choosing a single solution: it is easier to scope all of the work and minimizes the number of new skills required. On the other hand, mixing implementations may increase the amount of work required, but may accommodate your needs. Try testing different transition options to understand which one is best suited for your applets and needs. Regardless of the solution or solutions you decide upon, be aware of all planning issues and organizational impacts.

For most, a successful transition is one that results in “clean” systems without any MSJVM dependent files and applications, and with the MSJVM removed. However the definition of a “clean” system is based on your decision. If you are satisfied with simply removing the MSJVM, then once you have accomplished that goal you are finished. If you want to initiate a full scale migration project of every application with a MSJVM dependency to a .NET environment or install an alternate JRE, then once you have completed that goal you are done. At minimum your goal should be to identify MSJVM dependencies, and aside from any transition option you may choose, lock down security.

Remove Applets

Some applets do not add significant value to the web page, or the value added is not sufficient to justify developing replacements. If the applet can be removed without significant harm to the page, or if the entire page can be removed, consider removal as an option. This is done by removing the `<APPLET>...</APPLET>` text or commenting it out.

Advantages. Easy to implement, easy to test.

Drawbacks. Usually some loss of functionality on the page.

Deployment considerations. None beyond deployment of files.

Lock Down Security without Altering Applications

For some applications you may consider continuing to use the MSJVM. Legal licensees of Visual J++ 6.0 and the Microsoft SDK for Java may continue distribution of the MSJVM as part of the runtime for their application in accordance with the terms of their end user license agreement, or they may decide not to remove the MSJVM from locations where it is already installed. As mentioned earlier, leaving the MSJVM in place is not recommended as it may expose the system to potential security risks.

However, Internet Explorer (IE) contains security settings that can be used to disable or restrict the applicability of the MSJVM. You can choose to turn off the MSJVM for all websites. You can also elect to continue using the MSJVM but limit any future security issues by limiting use of Java to specific sites. After turning off Java in the Internet zone, depending upon the policy of your site, users or administrators can turn on Java for all trusted sites and then add the specific sites on which they want to support applets by including them in the Trusted Sites list.

Appropriate applications for this solution may be those that have low use or are near the end of their life cycles, and have a restricted user base or a user base that has no concern about possible future security issues. For example, an older application that will soon be obsolete but needs to be kept for future support or maintenance concerns might be isolated in a security lockdown.

Advantages. Inexpensive.

Drawbacks. Done on client machine, and therefore best suited for intranets; may reduce, but not completely eliminate, possible future security risks.

Skills requirement. Requires knowledge of organization's network security policies and procedures.

Deployment considerations. Security lockdown is on each end user's computer. Depending on your network configuration, IT may be able to set this as a Group Policy or propagate the settings through login scripts and .reg files.

Migrate or Rewrite Applications

Migrating or rewriting the code to another solution can often provide identical functionality to current applets and applications, and provides higher security. For example, migration to .NET allows you to take advantage of the power of the Windows platform. Applications in .NET are easy to extend and integrate well. By migrating to a .NET environment, support issues remain with Microsoft.

In some cases, business needs require a cross-platform solution where a .NET strategy is inappropriate. In this case, you have at least three options:

- DHTML
- Rewriting the application or applet for another display technology such as Flash or Shockwave
- Installing an alternate JRE

For more information regarding the migration or rewriting of applications refer to the *Microsoft Java Virtual Machine Migration Guide for Developers*.

Migrate to .NET and Visual J# (J# Browser Controls)

A Java applet or application can be upgraded to Visual J# by compiling it to a managed library using the Visual J# compiler (vjc.exe). If only the bytecode (.class files) exists for the Java component, the Visual J# Binary Converter Tool (JbImp.exe) can be used to convert the component to a managed library. When migrating an applet, the J# Browser Controls runtime is required in addition to the J# compilers.

Compiling a Java applet to a J# Browser Control usually does not require any changes to the Java applet source code. The J# Browser Control runtime provides support for functionality equivalent to most JDK 1.1.4 level packages, including the **java.applet** package.

Advantages. Microsoft supported conversion. Speed of implementation, and reuse of existing Java code.

Drawbacks. Automated conversion may still require some rewriting of code.

Skills requirement. Requires knowledge of .NET Framework and Visual J#.NET.

Deployment considerations. Migrated applets are installed on the Web server and called J# Browser Controls. The end user must install both the .NET runtime environment and the J# Browser Control runtime environment. Migrated applications behave just like regular applications, and require the .NET Framework and J# Redistributable to run.

Migrate to .NET and Visual C# (JLCA)

A Java applet or application can be rewritten in Visual C# using a tool such as the Java Language Conversion Assistant (JLCA). Often the JLCA converts more than 90% of the code automatically. The remaining code must be converted manually.

Advantages. Microsoft supported conversion. Access to the native .NET APIs and complete integration with .NET. The use of automated tools makes this a reasonably fast conversion. Applications converted are often faster and smaller than they were originally. Future extensions are possible that make use of the entire .NET Framework. Use of Windows forms and interface supports sophisticated user interfaces.

Drawbacks. Automated conversion still requires some rewriting of code.

Skills requirement. Requires knowledge of .NET Framework and Visual C#.

Deployment considerations. Migrated apps are installed on the Web server. The .NET runtime environment must be installed on each user's machine.

Use an Alternate Rendering Technology

Alternate rendering technologies including, DHTML, Flash, VB Script, Java Script, ECMA Script, etc. are also options. These technologies can implement most simple navigational items (for example, menus, rollovers, simple calculators, and chat clients). These are a good choice for converting low to medium complexity applets dedicated to navigation and user interface tasks.

Advantages. Significant speed advantages are achieved because the applet and JRE do not need to be loaded. Cross-platform compatibility is maintained.

Drawbacks. Not for rich applets or for extensive data manipulation and requires complete rewrite of applet code. Testing for each browser-platform combination can be time-consuming. Alternate rendering technologies are not supported by

Microsoft and may require you to establish new support contracts with the appropriate vendor.

Skills requirement. Requires knowledge of the appropriate coding language.

Deployment considerations. Migrated files are installed on the Web server. Availability to end users could be an issue for plug-ins and may require installation on each client.

Use an Alternate JRE

The Java applets or applications may run on an alternate JRE. Examine and test the possible JREs carefully because different virtual environments may implement different versions of the Java standard. If this is an external (Internet) facing site, each applet should contain a reference to where users can obtain the runtime. You may also want to make the choice to download the JRE explicit instead of referring to it in the <OBJECT> tag itself.

Advantages. Substantial Java code reuse, although some changes are still required for some applications.

Drawbacks. All systems need to download the selected JRE. Different JREs implement different versions of the Java standard. This solution increases complexity of support because problems must be routed through another vendor. If for some reason there are multiple Java runtime environments on a PC, you will need to explicitly point to the correct JRE and version to ensure that the applet runs correctly. Alternate JREs are not supported by Microsoft.

Skills requirement. Requires knowledge of Java implementations.

Deployment considerations. Web files may need to be updated to point to a new JRE; end users will need to upgrade their Java runtime environments.

Note Although Microsoft cannot vouch for the security and reliability of solutions offered by other companies, non-Microsoft solutions are available. Customers choosing to explore such solutions should engage in testing prior to pursuing this migration path.

4

Lock Down Security

IT professionals have the option of using Internet Explorer (IE) security settings to disable or restrict the applicability of the MSJVM. Locking down security rather than migrating to another solution allows the MSJVM to stay on some machines. This is a viable option for internal applications that are dependant on the MSJVM, but are not intended for public use. This security policy can be deployed to an organization as a Group Policy with Active Directory. However, if your organization currently does not utilize Active Directory, it will be necessary to make the appropriate adjustments on each client desktop.

This option reduces, but does not completely eliminate, security concerns with respect to the MSJVM. The optimal method for eliminating security concerns is to completely remove the MSJVM and dependencies, rather than limiting the execution of the MSJVM.

Locking Down Internet Explorer

In Internet Explorer, from the **Tools** menu, select **Internet Options** and then the **Security Settings** tab to disable the MSJVM. To turn off Java for the Internet zone, select the **Custom Settings** button. In the **Microsoft VM** section, under **Java Permissions**, select **Disable Java**. These settings are stored in the registry for the current user; they can be stored as a .reg file and propagated throughout an organization through a login script that loads the .reg file. On a machine with multiple user accounts, these changes will have to be done separately for each account. You should note that if you have installed Sun's Java plug-In for Windows on your system, this option also impacts the Sun JRE, which "impersonates" the MSJVM in IE.

A second possibility for customers who continue using the MSJVM is to limit security issues by limiting the use of Java to specific sites. After turning off Java in the Internet zone, they can turn on Java for all trusted sites and then re-enable support for applets by adding the specific sites with the applets to their Trusted Sites list.

Pointing to the MSJVM

An option when locking down the availability of the MSJVM to only secure and trusted sites is to install a second JRE. A second JRE allows applications that support multiple JREs, and may allow applets and applications without MSJVM-specific dependencies, to continue running. Test all JREs sufficiently during implementation to ensure the applications work properly with the newly installed JRE.

There are additional considerations if you are planning to install a second JRE on some machines that are locked down. For example, some JRE vendors provide a configuration

item that allows you to specify one JRE to run <OBJECT> tags and another JRE to run <APPLET> tags. Refer to the documentation associated with your JRE to determine how to configure tag handling.

The IE Security Zone settings specified applies to all <APPLET> tags. You can edit the Web files to use the <OBJECT> tag instead of the <APPLET> tag. The <OBJECT> tag is a more generic tag for embedded objects, including applets.

The structures of the two tags are similar:

```
<APPLET code=Applet1.class
  height=200 width=320
  name=Applet1
  VIEWASTEXT>
<PARAM NAME="foreground" VALUE="FFFFFF">
<PARAM NAME="background" VALUE="008080">
<PARAM NAME="label" VALUE="This string was passed from the HTML host.">
  No applet tag handlers installed.
</APPLET>
```

In the <OBJECT> tag, some attributes have become parameters (to avoid namespace conflicts), and there are additional parameters because the <OBJECT> tag is more generic:

```
<OBJECT CLASSID="clsid:08B0E5C0-4FCB-11CF-AAA5-00401C608501"
  height=200 width=320
  VIEWASTEXT>
<PARAM NAME="code" VALUE="Applet1.class">
<PARAM NAME="type" VALUE="application/x-java-applet">
<PARAM NAME="scriptable" VALUE="true">
<PARAM NAME="foreground" VALUE="FFFFFF">
<PARAM NAME="background" VALUE="008080">
<PARAM NAME="label" VALUE="This string was passed from the HTML host.">
  No Microsoft VM installed
</OBJECT>
```

The classid attribute is a URI that can indicate inline data. For an Active X control—and a JRE is considered an Active X control—the URI begins with “clsid” and is followed by the clsid value in the registry. The value 08B0E5C0-4FCB-11CF-AAA5-00401C608501 indicates the MSJVM; different versions of third-party JREs may use different numbers. Some Java plug-ins install on a Windows system by “impersonating” the MSJVM using the same registry entry and class ID on installation. Be cautious when installing JREs that impersonate the MSJVM, as you may need to configure the CLSID or the individual JREs to make sure the desired one loads.

If you are determined to point to a particular JRE in some circumstances, you may run into a second problem. As of this writing, at least one JRE installs a subkey under the key that identifies the MSJVM. The following key has as its default value the CLSID of the *new* JRE:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{08B0E5C0-4FCB-
11CF-AAA5-00401C608501}\TreatAs.
```

Attempting to explicitly load the MSJVM through the <OBJECT> tag when this key exists will actually load the new JRE. Deleting the subkey will solve this problem.

5

The MSJVM Removal Tool

The MSJVM Removal Tool allows you to completely remove the MSJVM. Removing the MSJVM is recommended for OEMs that are shipping versions of Windows on their systems and need to remove the MSJVM from their current image until migration to a new release of Windows that no longer contains the MSJVM is completed. It is also recommended for IT Professionals once the transition from dependency on the MSJVM is completed.

The tool completely removes the MSJVM with the exception of several registry entries that remain to ensure that other JREs, such as the Sun JRE, continue to work.

The removal option works with Windows 98, Windows 98 SE, Windows 2000, Windows Millennium Edition (ME), and Windows XP. Removal of MSJVM files are under system file protection in Windows 2000 and Windows ME; if you run the removal tool on one of these operating systems, all MSJVM files are replaced with 1-byte empty files of a higher version number, effectively disabling MSJVM.

It is important to test all transitioned applications in a test environment thoroughly prior to the removal of MSJVM. Removing the MSJVM is permanent and cannot be reversed. The options available to reinstall the MSJVM are to establish a system restore point, install a prior image of Windows that includes the MSJVM, or install a third party application that includes the MSJVM. Microsoft will not provide any copies of the MSJVM for installation.

To view a list of files, registry keys, and class libraries removed by the tool, please refer to: <http://www.microsoft.com/java>.

The MSJVM Removal Tool is available directly from Product Support Services. This utility has been pulled from Microsoft download servers to protect casual users from unintended, unexpected and typically irreversible effects that can result from misuse of the tool. To obtain a copy of the MSJVM Removal Tool, please refer to:

<http://support.microsoft.com/default.aspx?scid=kb;en-us;826878>

Note The following registry keys are not removed when the MSJVM is removed, this allows third-party JREs to function with Internet Explorer:

```
HKCR\Classes\CLSID\{08BOE5CO-4FCB-11CF-AAA5-00401C608501}
HKLM\SOFTWARE\Microsoft\Active Setup\Installed
Components\{08BOE5CO-4FCB-11CF-AAA5-00401C608500}
```

6

Switch to Another JRE

Note Microsoft does not guarantee or warrant that alternate JREs will work correctly in your environment. Test all JREs sufficiently during implementation to ensure that all of your needs are met.

Several vendors have announced that they will ship an alternate Java runtime environment (JRE) on their computers when Microsoft is no longer able to support the MSJVM. Whether you adopt this solution depends upon your business needs, including portability, performance, reliability, versions, and maintenance.

Although vendors represent that their JREs comply with relevant Java specifications, all JREs are not necessarily equivalent for your needs. Not all implementations of Java specifications are identical, and you should ensure that you test any alternate JRE prior to pursuing this option.

Current Visual J++ code may need to be ported to work with an alternate JRE. The use of Microsoft extensions is easy to find in current J++ source files; you can find Java source files that make use of the `@com` directive by using a CMD.EXE command of the form:

```
find /I "@com" c:\src\*.*
```

Similar commands will find other extensions, such as the `@dll` and `@security` directives. Once you have identified the use of the extensions, you can gauge the effects of removing them.

The use or removal of the Microsoft extensions makes no syntactic difference to the Java code because the directives are embedded in comments. It is possible to turn the extensions off with options to the `javac` command, so you can easily compare the effect of running the application with and without the extensions.

Alternate JREs are deployed throughout your organization using the same means by which you normally install new applications on user machines. For more information on properly installing an alternate JRE please refer to vendor documentation. If you choose to install an alternate JRE, any problems you experience with Java support will not be addressed by Microsoft, and are subject to relevant support and license agreements with the appropriate vendor of the alternate JRE.

The following is a partial list of available alternate JREs and is not intended to represent **all** available alternate JREs:

Sun JRE for Windows:

http://www.java.com/en/download/windows_automatic.jsp

IBM JRE (as part of the IBM WebSphere SDK for Web Services):
<http://www-106.ibm.com/developerworks/webservices/wsdk/>.

BEA JRockit:
<http://www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/jrockit>