



ServletExec™ 2.2

User Guide

for Microsoft Internet Information Server
Netscape FastTrack & Enterprise Server
the Apache Server
and Mac OS web servers

NEW ATLANTA COMMUNICATIONS, LLC

ServletExec™ 2.2 User Guide

September 29, 1999
version 2.2

Copyright © 1997-1999 New Atlanta Communications, LLC
1041-D Cambridge Square • Alpharetta, Georgia 30004-1871
Phone 678.366.3211 • Fax 678.366.3210
<http://www.newatlanta.com/>

ServletExec is a trademark of New Atlanta Communications, LLC

All other trademarks and registered trademarks herein are the property of their respective owners.

Table of Contents

1. GETTING STARTED.....	1
CHANGES FROM SERVLETEXEC 2.1.....	1
CHANGES FROM SERVLETEXEC 2.0.....	2
CHANGES FROM SERVLETEXEC 1.1.....	3
VERIFYING YOUR INSTALLATION	4
REGISTERING SERVLETEXEC	5
<i>Setting the Admin User Name & Password.....</i>	6
UNCONFIGURED SERVLETS	7
<i>Servlets with Multiple Class Files.....</i>	8
SERVLETEXEC DEBUGGER.....	8
TECHNICAL SUPPORT.....	9
WHAT NEXT?.....	9
2. SERVLETEXEC ADMINISTRATION	10
SERVLETEXEC	12
<i>Help.....</i>	12
<i>Register</i>	12
Setting the Admin User Name & Password.....	13
<i>About.....</i>	13
<i>View Logs.....</i>	13
SERVLETS	14
<i>Configure</i>	14
Invoking Servlets.....	15
Servlet Directories.....	16
Remote Servlets.....	16
Servlet Loading & Initialization	17
<i>Aliases.....</i>	18
Netscape WAI Aliases.....	19
Apache Aliases	19
Servlet Chains	20
The “invoker” Servlet.....	21
Servlets as Default Pages.....	21
<i>Filters.....</i>	23
<i>Logging</i>	23
SERVER-SIDE INCLUDES	24
<i>Counters.....</i>	24
<i>File Cache.....</i>	25
ADVANCED	25
<i>Virtual Servers</i>	25
Hardware & Software Virtual Servers.....	26
Microsoft IIS	26
Netscape FastTrack & Enterprise	26
Apache Server	26
Mac OS Web Servers	27
The “default” Server.....	27
Configuring a Virtual Server.....	27
Administering Virtual Servers.....	29
Compatibility with Older Browsers.....	30
<i>Security</i>	30
<i>VM Settings & Classpath</i>	31
Selecting a VM.....	31
Other VM Settings	33

Classpath	33
Session Tracking	34
IIS SECURITY	36
CONCLUSION.....	37
3. SERVER-SIDE INCLUDES (SSI)	38
SSI OVERVIEW	38
Netscape Enterprise SSI.....	38
WebSTAR SSI.....	39
CONFIGURING SSI.....	39
<SERVLET> TAG.....	40
SSI COMMANDS.....	41
Echo	42
Include.....	43
Fsize	43
Config.....	43
Show.....	44
Hide.....	45
Counter	45
Random	46
Nossi.....	46
4. SESSION TRACKING	47
SESSION TRACKING SUMMARY	47
SESSION TRACKING CONFIGURATION	48
SESSION TRACKING EXAMPLE	49
HTTPSESSION VALUE CLASSES.....	50
5. PRESENTATION TEMPLATES.....	51
PRESENTATION TEMPLATES SUMMARY	51
TEMPLATE SERVLET CONFIGURATION.....	52
USING PRESENTATION TEMPLATES WITH HTML PAGES	52
USING PRESENTATION TEMPLATES WITH SERVLETS	53
6. JAVASERVER PAGES.....	54
JAVASERVER PAGES SUMMARY.....	54
JAVASERVER PAGES CONFIGURATION.....	55
Configure the JSP10Servlet	55
Assign a Servlet Alias.....	56
Add tools.jar to the Classpath (JDK 1.2).....	56
Testing Your Configuration.....	57
USING JAVASERVER PAGES	57
<SERVLET> Tag.....	57
Invoking a JSP page from a Servlet	57
MICROSOFT IIS EXTENSION MAPPING	58
7. PAGE COMPILATION	60
PAGE COMPILATION SUMMARY	60
PAGE COMPILATION CONFIGURATION	61
Configure the JIServlet	61
Assign a Servlet Alias.....	62
Add tools.jar to the Classpath (JDK 1.2).....	62
Testing Your Configuration.....	62
USING PAGE COMPILATION.....	62
Page Compilation Example.....	63
KNOWN LIMITATIONS	63

8. FILE UPLOAD SERVLET	65
UPLOAD SERVLET CONFIGURATION.....	65
UPLOAD FEATURE SUMMARY	66
SERVLET CHAINING	66
KNOWN LIMITATIONS	67
APPENDIX A. MICROSOFT IIS/PWS INSTALLATION.....	1
UPGRADING FROM A PREVIOUS VERSION.....	1
SYSTEM REQUIREMENTS.....	2
UNINSTALL OTHER SERVLET ENGINES	3
WHAT WAS INSTALLED?	3
<i>The ServletExec ISAPI Directory</i>	3
<i>ServletExec_ISAPI.dll</i>	4
<i>Registry & Metabase Entries</i>	5
Filter DLLs Registry Entry.....	5
Metabase ISAPI Filter Entry	5
SERVLETEXEC ADMIN USER NAME & PASSWORD	7
JDK/JRE INSTALLATION	8
USER ACCOUNTS FOR MICROSOFT IIS	9
REINITIALIZING SERVLETEXEC.....	10
UNINSTALLING SERVLETEXEC.....	10
APPENDIX B1. NETSCAPE/NSAPI INSTALLATION (WINDOWS NT)	1
UPGRADING FROM A PREVIOUS VERSION.....	1
SYSTEM REQUIREMENTS.....	2
UNINSTALL OTHER SERVLET ENGINES	3
INSTALLING SERVLETEXEC FOR MULTIPLE SERVERS	3
DEACTIVATE THE JAVA INTERPRETER	4
WHAT WAS INSTALLED?	4
<i>The ServletExec NSAPI Directory</i>	4
<i>Registry Entries</i>	6
<i>Netscape Configuration File (obj.conf)</i>	6
<i>VerifyObjConf.bat</i>	7
JDK/JRE INSTALLATION	8
UNINSTALLING SERVLETEXEC.....	8
APPENDIX B2. NETSCAPE/NSAPI INSTALLATION (SPARC SOLARIS).....	12
UPGRADING FROM A PREVIOUS VERSION.....	12
SYSTEM REQUIREMENTS.....	13
UNINSTALL OTHER SERVLET ENGINES	14
INSTALLING SERVLETEXEC FOR MULTIPLE SERVERS	14
DEACTIVATE THE JAVA INTERPRETER	15
WHAT WAS INSTALLED?	15
<i>The ServletExecNSAPI Directory</i>	15
<i>Server “start” Script</i>	16
LD_LIBRARY_PATH.....	17
JNI_VERSION.....	17
LD_PRELOAD.....	17
SERVER_ROOT.....	18
<i>Netscape Configuration File (obj.conf)</i>	18
UNINSTALLING SERVLETEXEC.....	19
APPENDIX B3. NETSCAPE/WAI INSTALLATION & OPERATION	20
UPGRADING FROM A PREVIOUS VERSION.....	20
SYSTEM REQUIREMENTS.....	21
UNINSTALL OTHER SERVLET ENGINES	22

INSTALLING SERVLETEXEC FOR MULTIPLE SERVERS	22
DEACTIVATE THE JAVA INTERPRETER	22
ENABLE NETSCAPE WAI	23
WHAT WAS INSTALLED?	23
<i>The ServletExecWAI Directory</i>	23
<i>Netscape Configuration File (obj.conf)</i>	24
SERVLETEXECWAI OPERATION.....	25
<i>run<server name> Shell Script</i>	25
<i>Manual Startup (java Command)</i>	26
<i>Stopping ServletExecWAI</i>	27
REMOTE OPERATION.....	28
UNINSTALLING SERVLETEXEC.....	28
KNOWN LIMITATIONS	29
REFERENCES	29
APPENDIX C1. APACHE INSTALLATION & OPERATION (WINDOWS).....	1
SYSTEM REQUIREMENTS.....	1
UNINSTALL OTHER SERVLET ENGINES	2
MANUAL CONFIGURATION.....	2
<i>Prefix Aliases</i>	3
<i>Suffix Aliases</i>	3
WHAT WAS INSTALLED?	4
<i>The ServletExec Directory</i>	4
SERVLETEXECAPACHE OPERATION.....	6
<i>ServletExecApache.bat</i>	6
<i>Closing the DOS Window</i>	6
<i>Manual Startup (java Command)</i>	6
<i>Stopping ServletExecApache</i>	7
REMOTE OPERATION.....	8
MULTIPLE JAVA VMS	8
UNINSTALLING SERVLETEXEC.....	9
APPENDIX C2. APACHE INSTALLATION & OPERATION (UNIX)	10
SYSTEM REQUIREMENTS.....	10
UNINSTALL OTHER SERVLET ENGINES	11
MANUAL CONFIGURATION.....	11
<i>srm.conf</i>	11
<i>Prefix Aliases</i>	12
<i>Suffix Aliases</i>	12
<i>httpd.conf</i>	13
WHAT WAS INSTALLED?	13
<i>The servletexec Directory</i>	14
SERVLETEXECAPACHE OPERATION.....	15
<i>runapache</i>	15
<i>Manual Startup (java Command)</i>	15
<i>Stopping ServletExecApache</i>	16
REMOTE OPERATION.....	16
MULTIPLE JAVA VMS	17
UNINSTALLING SERVLETEXEC.....	18
APPENDIX D. MAC OS INSTALLATION	1
SYSTEM REQUIREMENTS.....	1
UNINSTALL OTHER SERVLET ENGINES	2
WHAT WAS INSTALLED?	2
UNINSTALLING SERVLETEXEC.....	2

APPENDIX E. DEVELOPING SERVLETS & MISCELLANY	1
SERVLETExec TECH SUPPORT FAQ	1
JAVA SERVLET API DOCUMENTATION	2
SERVLET THREADS	2
DEBUGGING SERVLETS	3
PROFILING SERVLETS.....	3
NATIVE METHODS	4
STATIC (CLASS) VARIABLES	4
SHARED CLASSES.....	4
HTTPSESSION VALUE CLASSES.....	4
SERVLET NAMES.....	4
(COMPILED CODE) IN STACK TRACES.....	5
USER ACCOUNTS FOR MICROSOFT IIS	5
UNIX FILE DESCRIPTORS	6



Getting Started

Even if You Don't Read Manuals, Please Read this Chapter!

This manual is for ServletExec 2.2, a Java™-based web application server that implements the Java Servlet API and JavaServer™ Pages (JSP) standards defined by Sun Microsystems. ServletExec 2.2 allows you to build web-based applications that can be deployed with Microsoft Internet Information Server (IIS & PWS) on Windows, Netscape FastTrack & Enterprise servers on UNIX and Windows NT, the Apache Server on UNIX and Windows NT, and Mac OS web servers such as WebSTAR and AppleShare IP.

ServletExec 2.2 implements the Java Servlet API as defined by the Java Servlet Development Kit (JSDK) 2.1, and the final JavaServer Pages (JSP) 1.0 specification.

This chapter will help you verify your ServletExec installation, register your copy of ServletExec, and explain the simplest method for invoking servlets. Finally, this chapter tells you how to contact us for technical support.

Changes from ServletExec 2.1

The following major enhancements were made in ServletExec 2.2:

- JavaServer Pages (JSP) 1.0 (p. 54)
ServletExec 2.2 implements the final JSP 1.0 specification.
- Microsoft VM (p. 31)
ServletExec 2.2 supports the Microsoft VM for Windows in a high-performance in-process configuration.

In addition to the major enhancements listed above, the following key changes were made from ServletExec 2.1 to 2.2:

- Microsoft IIS Extension Mapping (p. 58)
The Extension Mapping feature of Microsoft IIS 4.0 can be used to control access to JSP pages via NT File System (NTFS) security.
- Error Page per Virtual Server (p. 27)
A URL may be specified on the Virtual Servers admin page, which will be invoked when a ServletExec error occurs.

- Error Page for the SSI Servlet (p. 39)
A URL may be specified for the SSI Servlet, which will be invoked when errors occur during processing of SSI pages.
- Netscape Enterprise SSI (p. 38)
ServletExec 2.2 allows you to invoke servlets and JSP pages from within an NES server-side includes page.
- JDK 1.2 java.policy File
With ServletExec 2.2 it's no longer necessary to modify the java.policy file when using JDK 1.2.

Changes from ServletExec 2.0

The following major enhancements were made in ServletExec 2.1:

- Java Servlet Development Kit (JSDK) 2.1
ServletExec 2.1 implements version 2.1 of the Servlet API as defined by the JSDK.
- Sun HotSpot Performance Engine (p. 31)
ServletExec 2.1 implements support for Sun's HotSpot Performance Engine for JDK 1.2.x for improved performance.
- IBM Developer Kit for Windows, Java Technology Edition, Version 1.1.7 (p. 31)
ServletExec 2.1 implements support for IBM's Developer Kit for Windows, Java Technology Edition, Version 1.1.7 for improved performance.
- Internal Performance Enhancements
The internal ServletExec threading and security mechanisms were redesigned for improved performance. Already the fastest servlet engine available, ServletExec 2.1 is up to twice as fast as previous versions.
- ServletExec Debugger (p. 8)
The **FREE** ServletExec Debugger 2.1 is a companion to ServletExec that allows you to develop and debug servlets within your favorite Java IDE.
- Apache Multiple VM Support (pp. C-8, C-17)
For the Apache server, you can now run multiple ServletExec instances, each within its own Java VM, each associated with a different virtual host.

In addition to the major enhancements listed above, the following key changes were made from ServletExec 2.0 to 2.1:

- VM Settings (p. 31)
The Class GC and Async GC options can now be enabled/disabled.
- JSP and Page Compilation Configuration (pp. 55, 61)
It's no longer necessary to add Servlet20.zip to the classpath before using JSP or Page Compilation.

- Microsoft IIS User Accounts (p. A-9)
For Microsoft IIS, servlet requests are no longer processed under the SYSTEM account, but are run under the account of the authenticated user.
- Microsoft IIS Admin Users (pp. 6, 29)
For Microsoft IIS on Windows NT, it is no longer necessary to specify a password for the ServletExec Admin user, or for admin user for virtual servers. Instead, the Windows NT User password is used to authenticate access to the ServletExec Admin UI.
- Servlet Initialization Values (p. 17)
Servlet initialization argument values can now be enclosed in single- or double-quotation marks to allow the equals sign or comma characters in value strings.
- Upload Servlet Enhancements (p. 65)
ServletExec's built-in Upload Servlet now supports an initialization parameter to limit the size of uploaded files, and a request parameter to specify the name of the uploaded file.

Changes from ServletExec 1.1

The following major features were added in ServletExec 2.0:

- JavaServer Pages (p. 54)
ServletExec 2.0 implements the JavaServer Pages (JSP) standard based on Draft Specification 0.91. ServletExec will be upgraded to support the final JSP specification after it becomes available. This upgrade will be free to owners of ServletExec 2.0.
- Apache Server
ServletExec 2.0 supports the Apache Server 1.3.1 and greater on UNIX and Windows NT. See appendices C1 and C2 for complete installation instructions.

In addition to JavaServer Pages and Apache Server support, the following key changes were made from ServletExec 1.1 to 2.0:

- Allowed IP Addresses (p. 6)
Access to the ServletExec Admin UI can be restricted to specified IP addresses.
- View Logs Admin Page (p. 13)
ServletExec log files can be viewed remotely via the ServletExec Admin UI.
- Servlet Initialization Order (p. 17)
The initialization order of configured servlets can be specified.
- Servlets as Default Pages (p. 21)
Servlets can be invoked as default pages for directory index requests.
- Servlet Security (p. 30)
Set Factory, Print Job Access, and Clipboard Access have been added to servlet security settings.

- IIS Security (p. 36)
For Microsoft IIS, access to servlets and URLs that map to servlets can be restricted to authorized users.
- Upload Servlet in Servlet Chains (p. 66)
The Upload Servlet can be used in servlet chains. Servlets in the chain after the Upload Servlet can retrieve the request parameters.

Verifying Your Installation

Most of the ServletExec installers and installation scripts make all of the needed changes to your system to allow you to run servlets. Except for the Apache Server, if the ServletExec installer for your web server completed properly, there is no need for additional manual configuration. For the Apache Server, see appendices C1 and C2 for instructions for the manual configuration steps required to complete the ServletExec installation.

After running the ServletExec installer, restart your web server (if you're running ServletExec for Netscape/WAI or Apache, you must also start the ServletExecWAI or ServletExecApache application; see the appropriate appendix for instructions).

After restarting your web server you should verify your ServletExec installation using the included sample servlets. Invoke the sample servlets by entering the following URLs in your browser:

`http://www.yourserver.com/servlet/DateServlet`

`http://www.yourserver.com/servlet/TestServlet`

Replace `www.yourserver.com` in the above URLs with the host name or IP address of your server. (Note that the `/servlet/` virtual directory in the URL must be all lowercase and does not end with an "s". Also, **do not** configure a `/servlet/` virtual directory for your web server; if you do, ServletExec will not work properly).

If you're unable to successfully execute these sample servlets please review the installation information in Appendix A, B, C, or D (depending on your web server) to verify your installation. Also, check our on-line support FAQ for additional information:

<http://www.newatlanta.com/support-faq.html>

If necessary, contact us for help by sending email to the following address:

support@newatlanta.com

These examples illustrate the simplest way to invoke servlets. Because these servlets did not require any configuration we refer to them as "unconfigured servlets." The next section explains how to register your copy of ServletExec; the last section of this chapter discusses unconfigured servlets in more detail.

Registering ServletExec

You can register your copy of ServletExec by entering your serial number via the Administration User Interface (Admin UI). Access the ServletExec Admin UI via the following URL:

`http://www.yourserver.com/servlet/admin`

The upper portion of the Admin page accessed via this URL is illustrated in Figure 1.



The screenshot shows the top portion of the ServletExec Admin UI. It features a label 'Serial Number:' followed by a text input field. Below this is the text 'Mode: Demo'. A red message states: 'In Demo mode, ServletExec is fully functional for 100 requests. After processing 100 requests, ServletExec changes to Lite mode.' At the bottom of this section is a 'Register' button.

Figure 1. Registering ServletExec

Notice that prior to being registered, ServletExec operates in “Demo” mode. While in Demo mode, you have access to all ServletExec features, except that you won’t be able to enter the admin user name and password. This gives you an opportunity to “try before you buy.”

ServletExec will process 100 HTTP requests while in Demo mode, then switch to “Lite” mode. You can return to Demo mode by stopping and restarting your web server. There is no limit to the number of times you can run ServletExec in Demo mode. Running ServletExec in Demo mode is usually adequate for developing and debugging servlets.

While in Lite mode, the following ServletExec features are disabled:

- Servlet Aliases
- Servlet Chains
- Servlet Filters
- Remote Servlets
- Server-Side Includes (SSI), including the <SERVLET> tag
- Multiple Virtual Servers
- Session Tracking
- Page Compilation
- JavaServer Pages
- Presentation Templates
- File Upload Servlet
- IIS Security

Also, you won't be able to enter the admin user name and password while in Lite mode. ServletExec will process an unlimited number of requests while in Lite mode, so you can use it in this mode for free, forever!

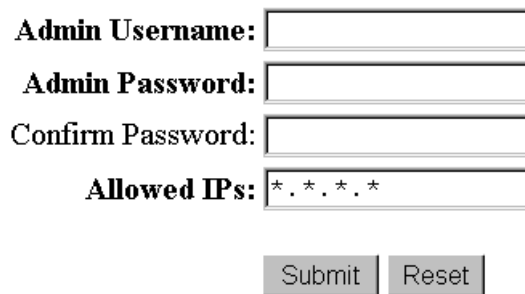
You can register ServletExec while in either Demo mode or Lite mode by entering your serial number and clicking "Register." After ServletExec is registered you won't be able to change the serial number. **YOU MUST PURCHASE A SEPARATE SERIAL NUMBER FOR EACH WEB SERVER ON WHICH YOU INSTALL SERVLETEXEC.** Do not reuse the same serial number on multiple web servers, as this would be a violation of the License Agreement.

Setting the Admin User Name & Password

After registering you should immediately assign the admin user name and password. This is done from the same ServletExec Admin UI page used to enter your serial number, as illustrated in Figure 2 on page 6 (for Microsoft IIS on Windows NT, the "Admin Password" and "Confirm Password" fields are not displayed; see further discussion below).

This user name and password will be required for all future access to the ServletExec Admin UI. For Microsoft IIS on Windows NT, the "Admin Password" and "Confirm Password" fields are not displayed. Instead, the "Admin Username" must be that of a Windows NT User. The password defined for the Windows NT User is used for authenticating access to the ServletExec Admin UI. Also, IIS 4.0 users must make sure that Basic Authentication is enabled (see Appendix A).

(If you forget your ServletExec Admin username and password, you can reset them by deleting the servers.properties file from the ServletExec Data directory. This file contains your ServletExec serial number and virtual server configuration data, in addition to the admin username and password. If you delete this file you'll need to re-enter your serial number and any virtual server configuration data you may have entered.)



The form contains four input fields and two buttons. The first three fields are for 'Admin Username', 'Admin Password', and 'Confirm Password'. The fourth field is for 'Allowed IPs' with a placeholder '*.*.*.*'. Below the fields are 'Submit' and 'Reset' buttons.

Admin Username:	
Admin Password:	
Confirm Password:	
Allowed IPs:	*. *.*.*.*
Submit Reset	

Figure 2. ServletExec Admin Username & Password

The Allowed IPs field allows you to specify a comma-separated list of IP addresses from which clients are allowed to access the ServletExec Admin UI. The specified IP addresses may include the "*" character to represent all IP addresses in a subrange. For example, the following list of IP addresses would allow access to the ServletExec Admin

UI from a client with the IP address 168.121.97.133, or any IP address in the range from 204.84.126.1 to 204.84.126.255:

`168.121.97.133, 204.84.126.*`

Access to the ServletExec Admin UI is always allowed from the local machine on which ServletExec is running regardless of the setting of the Allowed IPs field.

Unconfigured Servlets

Servlet class files are stored in the Servlets directory. The default location of the Servlets directory varies based on your operating system and web server. See the appropriate Appendix for your web server for the location of the Servlets directory. The name and location of the Servlets directory can be changed using the Configure Virtual Servers feature of the ServletExec Admin UI (see Chapter 2).

If you examine the Servlets directory after installing ServletExec you'll find two files: `DateServlet.class` and `TestServlet.class`. Notice that the names of these files, without the ".class" extension, are the same names we used to invoke the sample servlets in the URLs at the beginning of this chapter. This demonstrates how unconfigured servlets are invoked: place the class file for the servlet within the Servlets folder, and then invoke the servlet using a URL of the form:

`http://www.yourserver.com/servlet/ServletName`

where `www.yourserver.com` is the host name or IP address of your server, `/servlet/` is a virtual directory that ServletExec maps to the physical Servlets directory, and `ServletName` is the name of the servlet class file without the ".class" extension.

A servlet does not have to be configured if all of the following conditions are true:

- The class that implements the `doGet()`, `doPost()`, or `service()` methods is not part of a named Java package (the Java source file does not begin with a package statement).
- The servlet class files all reside on the local web server within the Servlets folder (see below for a discussion of servlets that consist of multiple class files).
- The servlet does not need to be initialized when the web server initializes.
- The servlet does not require initialization parameters.
- The servlet will never need to be manually unloaded.
- Only one instance of the servlet needs to be created.

If any of these conditions are not true, the servlet must be configured via the ServletExec Admin UI. See Chapter 2 for a discussion of configuring servlets.

Servlets with Multiple Class Files

If a servlet consists of multiple class files, the individual class files may be stored directly in the Servlets directory or within a compressed or uncompressed Java .jar/.zip archive within the Servlets directory.

If class files are placed directly in the Servlets directory, the directory structure must match the package structure of the classes. (Remember, in order to invoke an unconfigured servlet, the class that implements the `doGet()`, `doPost()`, or `service()` methods must not exist within a Java package and must reside directly within the Servlets directory. Other classes, however, are free to exist within Java packages).

If class files are placed within a Java .jar/.zip archive, the name of the archive must be the same as the name of the class that implements the `doGet()`, `doPost()`, or `service()` methods, plus the .jar/.zip extension. Classes within Java packages must be placed within .jar/.zip archives under a directory structure that matches the package structure. Java .jar/.zip archives must be placed directly within the Servlets directory and not within nested sub-directories.

Finally, classes that are shared by multiple servlets, or multiple instances of the same servlet, cannot be placed in the Servlets directory. Instead, such shared classes must be placed in a directory that appears in the ServletExec classpath (see Chapter 2 for a discussion of the ServletExec classpath).

See Chapter 2 for a more detailed discussion of the Servlets directory.

ServletExec Debugger

The best way to develop servlets that you plan to deploy on ServletExec is to use the ServletExec Debugger 2.2. The ServletExec Debugger is a basic web server that has the complete ServletExec servlet engine built in. The ServletExec Debugger allows you to develop and debug servlets in an environment that very closely simulates the ServletExec deployment environment.

The ServletExec Debugger can be used with almost any Java Integrated Development Environment (IDE) and comes with detailed instructions for use in the following popular Java IDEs:

- Symantec Visual Café 2.5 and 3.0
- Borland JBuilder 2 and 3
- Metrowerks CodeWarrior Pro 5
- IBM VisualAge for Java 2.0
- Sybase PowerJ 3.0

Support for additional IDEs is being added continuously, so check our web site to see if support for your favorite IDE has been added.

The ServletExec Debugger can be downloaded from our web site:

<http://www.newatlanta.com/downloads/index.html>

Best of all, the ServletExec Debugger is absolutely **FREE** for all users!

Technical Support

If you're having problems installing or using ServletExec, first check our on-line technical support FAQ:

<http://www.newatlanta.com/support-faq.html>

If you still need help after reviewing the FAQ, send us email describing the problem to:

support@newatlanta.com

Please indicate the operating system and web server you're using, and provide a complete description of the problem. You'll get a response within 24 hours (sooner in most cases). Be sure to include your phone number so we can call you if necessary.

Also, consider joining the ServletExec mailing list, a public discussion forum for servlet developers, and where we answer general questions about ServletExec.

To subscribe to the ServletExec mailing list, send email to:

list-requests@newatlanta.com

and include the following text in the message body:

`subscribe servletexec`

You'll receive a confirmation message with instructions on how to post messages to the list and how to unsubscribe.

What Next?

By this point you should have verified your ServletExec installation and registered your copy of ServletExec (or, you're running in Demo/Lite mode until you're ready to make a purchase). Don't forget to set the admin user name and password after registering!

You also know how to access the ServletExec Admin UI and invoke unconfigured servlets. The ServletExec Admin UI contains extensive online help and should contain all of the information you need to use ServletExec's features. However, if you prefer a lengthier discussion or printed documentation, Chapter 2 discusses each of ServletExec's features in detail and describes how to configure them via the ServletExec Admin UI.

ServletExec Administration

Using the ServletExec Admin User Interface (UI)

All ServletExec features are accessible from the administration user interface, known as the ServletExec Admin UI. You've already received an introduction to the ServletExec Admin UI if you've registered your copy of ServletExec as described in Chapter 1.

The ServletExec Admin UI contains extensive online help that duplicates much of the information contained in this chapter. After working with the ServletExec Admin UI for a short time you should be able to rely on the online help and may no longer need to refer to this manual.

Access the ServletExec Admin UI via the following URL (if you're running ServletExec for Netscape/WAI or Apache, you must start the ServletExecWAI or ServletExecApache application before you can access the ServletExec Admin UI; see Appendix B3, C1, or C2 for instructions):

`http://www.yourserver.com/servlet/admin`

where `www.yourserver.com` in the above URL with the host name or IP address of your server. If you haven't registered your copy of ServletExec the Register ServletExec page is displayed as a result of entering this URL. After registering, the Configure Servlets page is displayed for this URL.

The features of the ServletExec Admin UI are grouped into five major headings as displayed in the menu on the left side of the ServletExec Admin UI pages as illustrated by Figure 3 on page 11. These headings are:

- ServletExec – contains general features related to the overall operation of ServletExec including Help, Register, View Logs, and the About page.

- Servlets – contains features for configuring servlets, servlet aliases, servlet filters, and the servlet logging function.

- Server-Side Includes – contains features to configure ServletExec's built-in Server-Side Includes (SSI) functions such as page counters and the SSI file cache.

Advanced –contains advanced features such as configuring multiple virtual servers, servlet security, session tracking, classpath, and the Java Virtual Machine (VM) settings. This heading also contains the Shutdown feature for stopping the ServletExec WAI or ServletExec Apache application.

IIS Security – contains features to restrict access to servlets and URLs to specified Windows NT users. This feature is available only for Microsoft IIS.

The ServletExec Admin menu varies slightly based on your web server and operating system. Except for the IIS Security features, the differences are all in the options available under the Advanced heading; all other features (except IIS Security) are the same for all web servers and operating systems.

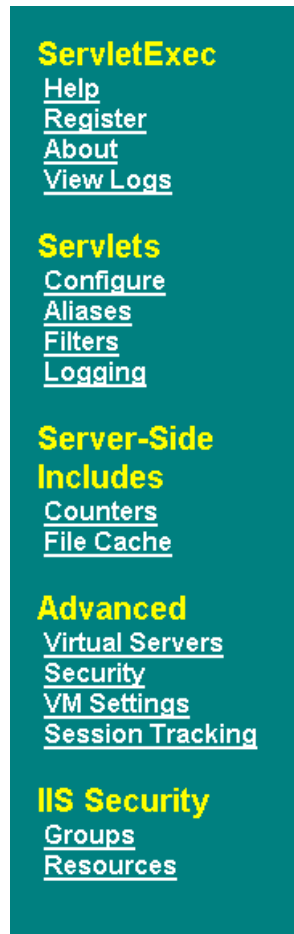


Figure 3. ServletExec Admin Menu for Microsoft IIS

ServletExec for Microsoft IIS and Netscape/NSAPI allows you to specify Java Virtual Machine (VM) configuration settings, including the classpath, using the VM Settings option. For ServletExec for Mac OS, the only Java VM setting that can be configured is the

classpath; therefore, under Advanced options, ServletExec for Mac OS has a selection for Classpath rather than VM Settings.

For ServletExec for Netscape/WAI and Apache, VM configuration settings are defined when the ServletExecWAI or ServletExecApache application is started (see Appendix B3, C1, and C2 for details). Therefore, the VM Settings page for ServletExec for Netscape/WAI and Apache only display information about the currently running VM, but do not allow modification of the VM settings. Also, the ServletExec for Netscape/WAI and Apache admin menus contain an additional option to Shutdown the ServletExecWAI or ServletExecApache application.

The ServletExec features can be accessed by clicking the hypertext links in the ServletExec Admin UI menu as illustrated by Figure 3 on p. 11. The sections below discuss each feature in detail.

ServletExec

The menu options under the ServletExec menu heading contains general features related to the overall operation of ServletExec including Help, Register, View Logs, and the About page.

Help

Each of the ServletExec Admin UI feature pages contains help text. In some cases, links to additional help pages are also provided. Therefore, this Help page simply contains a brief description of each feature page.

Register

This page allows you to enter your serial number to register your copy of ServletExec. It also allows you to set or change the admin user name and password, and the allowed client IP addresses for accessing the ServletExec Admin UI.

Prior to being registered, ServletExec operates in “Demo” mode. While in Demo mode, you have access to all ServletExec features, except that you won’t be able to enter the admin user name and password. This gives you an opportunity to “try before you buy.”

ServletExec will process 100 HTTP requests while in Demo mode, then switch to “Lite” mode. While in Lite mode, the following ServletExec features are disabled:

- Servlet Aliases
- Servlet Chains
- Servlet Filters
- Remote Servlets
- Server-Side Includes (SSI), including the <SERVLET> tag
- Multiple Virtual Servers
- Session Tracking
- Page Compilation
- JavaServer Pages

- Presentation Templates
- File Upload Servlet
- IIS Security

Also, you won't be able to enter the admin user name and password while in Lite mode. ServletExec will process an unlimited number of requests while in Lite mode, so you can use it in this mode for free, forever! In order to switch back to Demo mode from Lite mode you must restart your web server.

You can register ServletExec while in either Demo mode or Lite mode by entering your serial number and clicking "Register." After ServletExec is registered you won't be able to change the serial number. **YOU MUST PURCHASE A SEPARATE SERIAL NUMBER FOR EACH WEB SERVER ON WHICH YOU INSTALL SERVLETEXEC.** Do not reuse the same serial number on multiple web servers, as this would be a violation of the License Agreement.

Setting the Admin User Name & Password

After registering you should immediately assign the admin user name and password. This user name and password will be required for all future access to the ServletExec Admin UI. For Microsoft IIS on Windows NT, the "Admin Username" must be that of a Windows NT User. The password defined for the Windows NT User is used for authenticating access to the ServletExec Admin UI. Also, IIS 4.0 users must make sure that Basic Authentication is enabled (see Appendix A).

The Allowed IPs field allows you to specify a comma-separated list of IP addresses from which clients are allowed to access the ServletExec Admin UI. The specified IP addresses may include the "*" character to represent all IP addresses in a subrange. For example, the following list of IP addresses would allow access to the ServletExec Admin UI from a client with the IP address 168.121.97.133, or any IP address in the range from 204.84.126.1 to 204.84.126.255:

`168.121.97.133, 204.84.126.*`

Access to the ServletExec Admin UI is always allowed from the local machine on which ServletExec is running regardless of the setting of the Allowed IPs field.

About

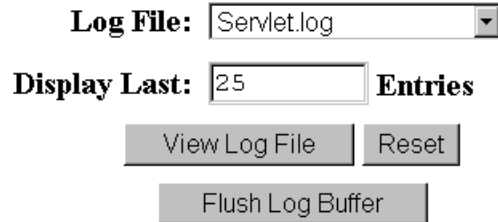
This page displays the ServletExec version number. Also contains information for signing up for the ServletExec mailing list (a discussion forum for servlet developers).

View Logs

This page allows you to view the contents of various log files created by ServletExec as illustrated in Figure 4. Select the log file you wish to view from the pop-up menu. These log files are described below.

ServletExec creates a log file named `ServletExec.log` in the ServletExec installation directory to which it writes informational and error messages. Output from calls to `System.out` and `System.err` made by servlets are also written to the `ServletExec.log` file.

When servlet logging is enabled (see Logging, p. 23) and a servlet invokes its `log()` method, the output is written to the `Servlet.log` file in the Servlet Logs directory. Note that ServletExec buffers output to the `Servlet.log` file; use the Flush Log Buffer button to immediately write the contents of the buffer to the log file.



The screenshot shows a web interface for viewing logs. It includes a 'Log File' dropdown menu currently set to 'Servlet.log'. Below it is a 'Display Last' input field containing the number '25', followed by the word 'Entries'. At the bottom of the form are three buttons: 'View Log File', 'Reset', and 'Flush Log Buffer'.

NOTE: the `Servlet.log` file may have entries cached in memory so you should flush the log buffer before viewing this log file.

Figure 4. View Logs Admin Page

For Microsoft IIS and Netscape FastTrack & Enterprise on Windows, ServletExec creates a log file named `ServletExecNative.log` in the ServletExec installation directory to which it writes informational and error messages generated by the native code portions of ServletExec.

Servlets

The menu options under the Servlets heading contain features for configuring servlets, servlet aliases, servlet filters, and the servlet logging function.

Configure

In some cases ServletExec can execute servlets without being configured. However, in many cases it's necessary to configure servlets before they can be invoked. Servlets **must** be configured if any of the following conditions are true:

- The class that implements the `doGet()`, `doPost()`, or `service()` methods is part of a named Java package (if the Java source file begins with a package statement).
- The servlet classes are to be loaded from a remote web server.
- The servlet needs to be initialized when the web server initializes.
- The servlet requires initialization parameters.
- The servlet may need to be manually reloaded.
- More than one instance of the servlet needs to be created.

If none of the above conditions are true for a servlet, the servlet can be invoked without being configured. *For unconfigured servlets, the Servlet Name is the name of the class that implements the `doGet()`, `doPost()`, or `service()` methods.* Use this class name

wherever you would use the Servlet Name in the following discussion (also see the discussion of unconfigured servlets in Chapter 1).

Figure 5 shows the configuration of a fictional servlet, as it would appear in the ServletExec Admin UI. This figure will be used to illustrate the following discussion. Configured servlets are listed on the configuration page in alphabetical order based on Servlet Name.

The screenshot shows a web form for configuring a servlet. It contains the following fields and controls:

- Servlet Name:** A text input field containing the value "Foobar".
- Servlet Class:** A text input field containing the value "newatlanta.servlets.FoobarServlet".
- Code Base:** An empty text input field.
- Initialization Arguments:** A text area containing the value "foo=true, bar=false".
- Init Load Order:** A small text input field containing the value "1".
- Loaded:** A checkbox that is checked, followed by the text "Loaded".
- Buttons:** Two buttons labeled "Submit" and "Reset" are located at the bottom right of the form.

Figure 5. Servlet Configuration Admin Page

Invoking Servlets

Servlets can be invoked by specifying the Servlet Name in a URL or within the `<SERVLET>` tag of a server-side includes (SSI) page (refer to the SSI documentation in Chapter 3 for a description of the `<SERVLET>` tag). When invoking servlets via URLs, precede the Servlet Name with the `/servlet/` virtual directory. For example, the following URL invokes the servlet named Foobar:

```
http://www.newatlanta.com/servlet/Foobar
```

The `/servlet/` virtual directory, Servlet Name, and Servlet Class are case-sensitive. In this example, the `doGet()` or `service()` method of the Servlet Class corresponding to Foobar gets invoked. In this example the Servlet Class is:

```
newatlanta.servlets.FoobarServlet
```

Notice that the Servlet Class specifies the full name of the class including package information. Also, notice that the mapping of Servlet Names to Servlet Classes is arbitrary; that is, there are no rules for defining Servlet Names based on the value of Servlet Class.

Again, `/servlet/` in the above URL is a virtual directory that gets mapped to a physical directory by ServletExec when it looks for the Servlet Class (see the following discussion on Servlet Directories).

One instance of the Servlet Class is created for each Servlet Name configured for that class. Multiple Servlet Names may be configured for a Servlet Class to create multiple instances of that class; this may be useful if servlet behavior is controlled by initialization arguments because different instances of a Servlet Class could be given different initiali-

zation arguments. The instance that gets invoked is determined by the Servlet Name specified in the URL.

Servlet Directories

Servlet classes can reside on the local web server or a remote web server. By default, local classes are placed in the Servlets directory. This is the directory to which the /servlet/ virtual directory gets mapped for local servlets. (**Microsoft IIS users:** *do not* create a “servlet” virtual directory! ServletExec performs this mapping internally. If you create an IIS virtual directory named “servlet” it will interfere with ServletExec’s operation).

The default location of the Servlets directory varies based on your operating system and web server. See the appropriate Appendix for your web server for the location of the Servlets directory. The name and location of the Servlets directory can be changed using the Configure Virtual Servers feature of the ServletExec Admin UI (see below).

Servlet classes can be stored in Java .jar/.zip archives within the Servlets directory, or the individual .class files can be placed in the Servlets directory. In the latter case, the directory structure within the Servlets directory must reflect the package structure of the classes. For example, the directory structure for the servlet class newatlanta.servlet.FoobarServlet would appear as follows:

```
Servlets
|
+-newatlanta
|
|   +-servlet
|   |
|   |   +FoobarServlet.class
```

If a servlet’s classes are stored in a Java .jar/.zip archive, the archive file name must be the same as the name of the class that implements the doGet(), doPost(), or service() methods, excluding package information. For example, the .jar archive for newatlanta.servlet.FoobarServlet would be named FoobarServlet.jar and placed directly in the Servlets directory:

```
Servlets
|
+-FoobarServlet.jar
```

Finally, classes that are shared by multiple servlets, or multiple instances of the same servlet, cannot be placed in the Servlets directory. Instead, such shared classes must be placed in a directory which appears in the ServletExec classpath (see the heading VM Settings, below, for a discussion of the ServletExec classpath).

Remote Servlets

For servlet classes that reside on remote web servers, the Code Base parameter specifies the location of the directory or .jar/.zip archive that contains the servlet classes. For example:

`http://www.newatlanta.com/coolervlets/`

`http://www.newatlanta.com/servlets/coolervlet.jar`

The directory specified by the Code Base parameter is the directory to which the `/servlet/` virtual directory gets mapped for remote servlets. The Code Base parameter is not used for local servlets.

Remote servlets are always untrusted and therefore have the following restrictions:

- cannot use Java networking classes
- can only read a limited set of System properties
- cannot invoke an external (native) process
- cannot load a dynamic (native) library
- cannot create a class loader
- cannot perform any operations on files
- cannot set stream and socket handler factories
- cannot initiate print job requests
- cannot access the system clipboard

Because of these security restrictions, remote servlets must exist as a single `.class` file or `.jar/.zip` archive. A security exception will occur if a remote servlet exists in multiple `.class` files.

Servlet Loading & Initialization

Initialization arguments are passed to the servlet's `init()` method when the servlet is loaded. Initialization Arguments are specified as comma-separated name-value pairs; the "value" must be enclosed in single- or double-quotation marks if the value contains either the equals sign (=) or comma (,) characters. Otherwise, the quotation marks are optional. For example:

`name1=value1, name2="value2, in quotes", name3=value3`

Normally, servlets are loaded and initialized the first time they're invoked. In some cases it may be desirable to load servlets when ServletExec initializes (when the web server initializes). For example, a servlet may create a background thread that needs to be running before the servlet is invoked the first time.

The Init Load Order parameter takes a numeric value that specifies the order in which servlets are loaded during ServletExec initialization. If this parameter is left blank the servlet is not loaded during ServletExec initialization, but will be loaded the first time it's invoked.

The Loaded checkbox indicates whether a servlet is currently loaded. Servlets can be manually loaded by checking the Loaded checkbox (and submitting the form). Servlets can be manually unloaded by unchecking the Loaded checkbox (and submitting the form). Unloading servlets manually can be useful when modifications are made to remote servlet classes, or when changing servlet initialization arguments.

Local servlet classes are examined for modifications whenever the servlet is invoked; the new classes are loaded if changes are detected (even if the class files reside in a Java .jar/.zip archive). Remote servlet classes are not checked for modifications when invoked because this could cause unacceptable performance degradation due to networking delays. Manually unloading a remote servlet will cause the classes to be reloaded the next time the servlet is invoked. In this manner, ServletExec can be forced to reload modified remote servlet classes.

Aliases

Servlet aliases provide an alternative to the `/servlet/ServletName` method of invoking servlets. Servlet Aliases also provide a mechanism for specifying **servlet chains** for request processing (servlet chains are discussed in more detail below). Figure 6 illustrates the configuration of several servlet aliases and serves as a reference for the remainder of this discussion.

The Servlet Name(s) with which a Servlet Alias is associated must be configured using the Configure Servlets page as described above.

Servlet aliases are case-sensitive and exist in two forms: suffix and prefix. A suffix alias is one that begins with “*”; any URL that ends with the suffix invokes the specified servlet or servlet chain.

Alias	Servlet Name(s)
/date	DateServlet
/sv	invoker
*.foobar	FooServlet,BarServlet
*.jhtml	JIServlet
*.shtml	SSIServlet

Figure 6. Servlet Aliases

For example, in Figure 6, both “*.shtml” and “*.foobar” define suffix aliases. The “*.shtml” alias is defined by the default ServletExec configuration and causes all URLs ending with the “.shtml” extension to be processed by ServletExec’s internal SSIServlet. (Note that you can change the extension used by ServletExec for SSI file processing by changing this default suffix alias). With this configuration, the following URLs will be sent to ServletExec’s internal SSIServlet for processing:

```
http://www.newatlanta.com/ssiexample.shtml
http://www.newatlanta.com/ssiexample.shtml?query=args
```

```
http://www.newatlanta.com/some/path/myfile.shtml
```

As illustrated by the “*.shtml” and “*.foobar” examples, suffix aliases are often used to implement “file extension mapping.” That is, any file with the specified extension is mapped to the specified servlet or servlet chain. However, there is no requirement to limit use of suffix aliases to file extensions.

Any alias which is not a suffix alias (does not begin with “*”) is automatically a prefix alias. In Figure 6, “/date” is a prefix alias; any URL beginning with this prefix is routed to the DateServlet for processing, for example:

```
http://www.newatlanta.com/date
http://www.newatlanta.com/dateoftoday
http://www.newatlanta.com/date/today.html
```

Note that if you don’t include the path separator, “/”, when configuring a prefix alias then ServletExec adds it automatically.

If a URL matches both a prefix and suffix alias, only the servlet or servlet chain associated with the prefix alias is invoked (prefix aliases are checked before suffix aliases). The order in which URLs are compared to servlet aliases is the order in which they’re listed on the ServletExec Admin UI page as illustrated in Figure 6.

Netscape WAI Aliases

For ServletExec for Netscape/WAI, in addition to configuring servlet aliases via the ServletExec Admin UI, you must also add a NameTrans directive to the Netscape **obj.conf** configuration file for each alias. The NameTrans directives for the aliases in Figure 6 would appear as follows:

```
NameTrans fn="assign-name" from="/date*" name="servletexec"
NameTrans fn="assign-name" from="/sv*" name="servletexec"
NameTrans fn="assign-name" from="*.foobar" name="servletexec"
NameTrans fn="assign-name" from="*.shtml" name="servletexec"
NameTrans fn="assign-name" from="*.jhtml" name="servletexec"
```

Note that for suffix aliases, the “from” parameter of the NameTrans directive takes the same form as the alias as configured in the ServletExec Admin UI. For prefix aliases, the “from” parameter includes an additional trailing asterisk (“*”).

See Appendix B3 for more information regarding the **obj.conf** file for Netscape/WAI.

Apache Aliases

(Note: beginning with Apache 1.3.4, use of the **srn.conf** configuration file has been deprecated, and all configuration directives are now placed in **httpd.conf**).

For ServletExec for Apache, in addition to configuring servlet aliases via the ServletExec Admin UI, you must all add directives to the Apache **srn.conf** configuration file for each alias. For each suffix alias, an AddHandler directive must be added to **srn.conf** in the following format:

```
AddHandler servlet-exec <suffix-alias>
```

For each prefix alias, a Location directive must be added to **srm.conf** in the following format:

```
<Location /prefix-alias>
    SetHandler servlet-exec
</Location>
```

The Apache **srm.conf** directives for the servlet aliases in Figure 6 would appear as follows:

```
<Location /date>
    SetHandler servlet-exec
</Location>

<Location /sv>
    SetHandler servlet-exec
</Location>

AddHandler servlet-exec foobar
AddHandler servlet-exec shtml
AddHandler servlet-exec jhtml
```

See Appendices C1 and C2 for more information regarding the Apache **srm.conf** file and configuring servlet aliases.

Servlet Chains

Servlet chains can be configured to allow multiple servlets to process a request. Servlet chains are defined by specifying multiple Servlet Names, separated by commas, for a servlet alias. In Figure 6, the “*.foobar” suffix alias is associated with a servlet chain. Requests for URLs that end with the “.foobar” file extension are first processed by FooServlet. The output stream from FooServlet is directed to the input stream of BarServlet, which then produces the final output stream that is sent to the client.

Generally, servlets must be specifically designed by the servlet developer to participate in servlet chains. The exception is the first servlet in the chain, which is not required to do any special processing to participate in the chain. Servlets that are not first in the servlet chain must read the output stream of the previous servlet in the chain.

(Servlet Developers: the “output stream of the previous servlet in the chain” is read from a ServletInputStream object, which is retrieved by invoking the `getInputStream()` method on the `HttpServletRequest` or `ServletRequest` object passed as a parameter to your servlet’s `service()`, `doGet()`, or `doPost()` method.)

One possible use of servlet chains is to implement a complex servlet as a chain of modular servlets. Another possible use is to implement a “post-processing” servlet that could be chained to any other servlet (this specific, or per-chain post-processing is in contrast to the generic post-processing facility provided by Servlet Filters, discussed in the next section).

The “invoker” Servlet

ServletExec recognizes a special Servlet Name that can be used to provide alternatives to the `/servlet/` virtual directory used in URLs to invoke servlets (see the heading *Invoking Servlets*, above). The Servlet Name `invoker` may be mapped to prefix aliases as illustrated in Figure 6. The name `invoker` must be all lowercase, and the prefix alias must not end with `/`. In this case, the virtual directory `/sv/` may now be used in URLs to invoke servlets. For example:

```
http://www.newatlanta.com/sv/TestServlet
```

The `invoker` cannot be used in servlet chains.

Servlets as Default Pages

ServletExec allows you to invoke servlets as default pages for a web server based on suffix aliases. For example, if the default page for the web server is configured as `index.shtml`, ServletExec will translate directory index requests (URLs ending with `"/`) to `/index.shtml` and then forward the request to the `SSIServlet` (because of the `*.shtml` suffix alias).

Generally, there must be a file that matches the default page name in the directories for which this feature is enabled. For example, assume a suffix alias `*.dtm` is defined for the `DateServlet` and a default page `index.dtm` is specified for the server. In this case a file named `index.dtm` must be placed in directories for which the `DateServlet` is to be invoked as the default page. The contents of the `index.dtm` file are irrelevant.

Invoking servlets as default pages in this manner is supported for the following web servers (with limitations, if any, described below):

- Netscape FastTrack & Enterprise for Solaris (NSAPI)
- Netscape FastTrack & Enterprise for Windows (NSAPI)
- Microsoft IIS
- Apache Server

Invoking servlets as default pages is not supported for the following web servers:

- Netscape Enterprise/WAI
- MacOS web servers

Details of how this feature is implemented for different web servers, and their limitations (if any), are described in the following sections.

Netscape FastTrack & Enterprise (NSAPI)

With Netscape FastTrack & Enterprise (NSAPI) servers, ServletExec determines the default pages by extracting them from the following directive in the **obj.conf** file:

```
PathCheck fn="find-index" index-names="<default1>,<default2>"
```

When ServletExec receives a request for a directory index (a request URL ending with “/”) it searches the directory for a default page in the order in which the default pages are listed in the PathCheck directive in **obj.conf**. If ServletExec finds a default page that maps to a servlet via a servlet alias, then the servlet is invoked to process the request.

When the default pages in the PathCheck directive in **obj.conf** are modified, the web server must be restarted for ServletExec to pick up the changes.

Microsoft IIS 3.0

Microsoft IIS 3.0 allows only one default page to be configured. When ServletExec receives a request for a directory index (a request URL ending with “/”), if the default page maps to a servlet via a servlet alias, then the servlet is invoked to process the request.

The known limitations for invoking servlets as default pages with IIS 3.0 are:

1. When the IIS 3.0 default page setting is modified, ServletExec must be re-initialized in order for it to pick up the change. For IIS 3.0 this requires stopping all of the IIS services (FTP, WWW, Gopher) and then re-starting them.
2. If the default page maps to a servlet and a request is received for a directory that doesn't contain a default page, the servlet will be invoked to process the request anyway.

Microsoft IIS 4.0

Microsoft IIS 4.0 keeps a master list of default pages for all web sites and virtual directories, and also allows each web site and virtual directory to have its own list of default pages. ServletExec only uses the first default page in the master list of default pages for invoking servlets. When ServletExec receives a request for a directory index (a request URL ending with “/”), if the first default page in the master list of default pages maps to a servlet via a servlet alias, then the servlet is invoked to process the request.

The known limitations for invoking servlets as default pages with IIS 4.0 are:

1. When the default page setting is modified, ServletExec must be re-initialized in order for it to pick up the change. For IIS 4.0 this requires stopping the IIS Admin Service from the services control panel and then re-starting the web server.
2. If the default page maps to a servlet and a request is received for a directory that doesn't contain a default page, the servlet will be invoked to process the request anyway.
3. The first default page in the master list of default pages is used for all web sites and virtual directories.

Apache Server

(Note: beginning with Apache 1.3.4, use of the **srn.conf** configuration file has been deprecated, and all configuration directives are now placed in **httpd.conf**).

Default pages for the Apache Server are defined using the `DirectoryIndex` variable in the `srm.conf` file. When ServletExec receives a request for a directory index (a request URL ending with “/”) it searches the directory for a default page in the order in which the default pages are listed in the `DirectoryIndex` variable in `srm.conf`. If ServletExec finds a default page that maps to a servlet via a servlet alias, then the servlet is invoked to process the request. There are no limitations for invoking servlets as default pages with the Apache Server.

Filters

Servlet filters provide a facility for doing response post-processing based on the MIME type of the response. For example, in Figure 7 the servlet named FooBar is configured to filter MIME type “foo/bar”. Any output response stream with MIME type “foo/bar” produced by any servlet is redirected to the input stream of servlet FooBar.

(Servlet Developers: the “output response stream” is read from a `ServletInputStream` object, which is retrieved by invoking the `getInputStream()` method on the `HttpServletRequest` or `ServletRequest` object passed as a parameter to your servlet’s `service()`, `doGet()`, or `doPost()` method.)

Mime Type	Servlet Name
foo/bar	FooBar
Submit Reset	

Figure 7. Servlet Filters

Before being configured as a servlet filter, the Servlet Name must be configured using the Configure Servlets page as described above. Only one servlet filter can be configured per MIME type and servlet chains cannot be used as servlet filters. Also, MIME types are case-sensitive.

Only one filter is applied per response. That is, if a servlet that is acting as a filter produces a response with a MIME type that is also mapped to a servlet filter, the second filter is not applied. This is done to prevent infinite loops of filters. For example, if the FooBar servlet produces a response with MIME type of “foo/bar” that response will not be redirected back to the FooBar servlet a second time.

Logging

When servlet logging is enabled and a servlet invokes its `log()` method, the output is written to a file named `Servlet.log` in the Servlet Logs directory. For virtual servers, sub-directories are created within the Servlet Logs directory for each virtual server. Servlet logging can be enabled or disabled from the ServletExec Admin UI as illustrated in Figure 8 on page 24.

Servlet log entries are buffered in memory by ServletExec and only written to the `Servlet.log` file after the buffer limit is exceeded. The default buffer limit is 40 entries. The Flush Log Buffer button causes ServletExec to immediately write the contents of the buffer to the log file.

ServletExec will continue writing entries to the `Servlet.log` file until the log rollover limit, as specified in K bytes, is reached. When the log rollover limit is reached, the `Servlet.log` file is renamed `Servlet.log.x`, where “x” is incremented by 1 for each rollover, and a new `Servlet.log` file is created. The default log rollover limit is 100K bytes.

Servlet Logging: ☒ Enabled ☐ Disabled

Servlet Log Buffer Limit: Log Entries

Servlet Log Rollover Limit: K Bytes

Figure 8. Servlet Logging

Server-Side Includes

The menu options under the Server-Side Includes heading includes features for configuring the page counter and file cache options of ServletExec’s built-in Server-Side Includes (SSI) servlet. Chapter 3 of this manual contains a detailed description of ServletExec’s built-in SSI features.

Counters

Page counters are created using the `#counter` SSI command. The ServletExec Admin UI displays the name of each page counter, its current value, and the date and time it was last reset to 0 (zero) as illustrated in Figure 9 on page 25. A page counter can be manually reset to zero by clicking the “Zero” button associated with the counter. A page counter can be removed by clicking the “Remove” button associated with the counter.

Removing a page counter has a similar effect as resetting it to zero. That is, if a page counter is removed, when the SSI page referencing that counter is next served the counter will be recreated with a value of 1 (one). A page counter should be removed when references to it are removed from SSI pages (or the SSI pages referencing the counter are removed from the web site).

SSI Counters

Name	Value	Last Zeroed	Commands
sitcounter	4	Tue Jun 16 09:09:30 EDT 1998	Zero Remove

Figure 9. SSI Counters

File Cache

For performance reasons, ServletExec stores SSI pages in memory as they're referenced. The default size of the file cache is 100K bytes. The size of the file cache can be changed by entering the new value (measured in K bytes) and clicking the Submit button.

When an SSI page is modified the file cache should be flushed so that the new page will be read from disk the next time it's referenced. Otherwise, ServletExec will continue to use the old file from its cache. Clicking the "Flush Cache" button will flush the SSI file cache.

The SSI file cache should be disabled when developing SSI pages so that the latest version of the SSI page will always be used. Re-enable the SSI file cache after the SSI pages are finished.

File Cache: ☒ Enabled ☐ Disabled

File Cache Size: K Bytes

Figure 10. SSI File Cache Settings

Advanced

The menu options under the Advanced heading includes features for configuring multiple virtual servers, servlet security, and Java VM settings (including the classpath variable).

Virtual Servers

"Multi-hosting" is a web server feature whereby a single physical server hosts multiple virtual servers with different domain, or host, names. For example, a single physical server may host both "www.abc.com" and "www.xyz.com".

In multi-hosting environments, ServletExec configuration options can be set separately for each virtual server. Additionally, a separate admin user name and password, and allowed client IP addresses for accessing the ServletExec Admin UI can be defined for each virtual server.

Hardware & Software Virtual Servers

Virtual server implementations come in two forms. One form, known as “Hardware Virtual Servers” or “IP-based Virtual Servers” assigns a different IP address to each virtual server. In the other form, known as “Software Virtual Servers,” the physical server has only a single IP address. The main effect on ServletExec of the form of virtual server implementation in use is the manner in which the document root directory is defined for each virtual server.

Most of the configuration for ServletExec to support multiple virtual servers is done via the ServletExec Admin UI; further details are provided below. For Netscape and Apache servers, additional directives may be needed in the **obj.conf** or **httpd.conf** configuration files, respectively. See the appropriate appendices for additional discussion of these configuration files.

Microsoft IIS

Microsoft IIS only supports virtual servers on Windows NT Server; virtual servers are not supported on Windows NT Workstation or Windows 95/98. IIS 3.0 only supports hardware virtual servers; IIS 4.0 supports both hardware and software virtual servers.

The document root directory is specified for each virtual server during IIS configuration. For each IIS virtual server, you must define a virtual directory that maps to the physical directory that contains the ServletExec_ISAPI.dll file. The virtual directory must have execute permission enabled. For the default web site, this is the SCRIPTS virtual directory, which maps to C:\InetPub\Scripts.

Netscape FastTrack & Enterprise

Netscape web servers support both hardware and software virtual servers. For hardware virtual servers, a separate NameTrans entry must be added to the **obj.conf** file for each virtual server. See Appendices B1 and B2 for discussions of the **obj.conf** file. For software virtual servers, the web server document root is the document root directory for all virtual servers; no additional configuration is required for ServletExec.

In addition to supporting multiple virtual servers, Netscape web servers allow you to install multiple *instances* of the web server. Each server instance is associated with a unique IP address and/or port. You must install ServletExec separately for each Netscape server instance, and the server instances (and ServletExec) are configured independently, as if each were a standalone server. (Note that when using this configuration, you must purchase a unique ServletExec serial number for each Netscape server instance with which ServletExec will be installed).

Apache Server

The Apache server supports both hardware and software virtual servers. In either case, virtual servers are configured using <VirtualHost> directives within the **httpd.conf** configuration file. See Appendices C1 and C2 for discussions of directives within the **httpd.conf** file as they relate to servlet aliases and virtual servers.

When the ServletExecApache application is launched, the root document directory for each virtual server must be specified. See Appendices C1 and C2 for further discussion.

In addition to supporting multiple virtual servers, the Apache server allows you to install multiple *instances* of the web server (referred to as multiple *daemons* in the Apache manual). Each server instance is associated with a unique IP address and/or port. You must install ServletExec separately for each Apache server instance, and the server instances (and ServletExec) are configured independently, as if each were a standalone server. (Note that when using this configuration, you must purchase a unique ServletExec serial number for each Apache server instance with which ServletExec will be installed).

Mac OS Web Servers

Web servers running on Mac OS 8.0 or earlier with Open Transport 1.2 or earlier can only support software virtual servers; web servers running on Mac OS 8.1 or later with Open Transport 1.3 or later can support both software and hardware virtual servers. Mac OS web servers implement virtual servers via plug-ins provided by the server vendor or third parties. For a list of third-party plug-ins see:

<http://www2.starnine.com/extendingwebstar/multidom.tmp1>

These multi-hosting plug-ins require you to configure the document root directory for each virtual server.

For plug-ins that implement version 1.2 of the WebSTAR API (W*API), you must configure ServletExec to use the same document root directory as configured for the multi-hosting plug-in for each virtual server. For plug-ins that implement version 1.3 of the W*API, configure the document root directory for every virtual server as “:” (as it is for the “default” server). Consult your plug-in vendor to determine which version of the W*API it implements.

The “default” Server

The configuration options for a server named “default” are created automatically by ServletExec the first time it initializes. ServletExec uses the configuration options for the default server when an HTTP request is received for an unconfigured virtual server.

For Windows and UNIX web servers it’s not necessary to configure all (or any) virtual servers; ServletExec will use the configuration options for the default server for unconfigured servers. For Mac OS web servers and multi-hosting plug-ins that implement version 1.2 of the W*API, you *must* configure **all** virtual servers in multi-hosting environments. This is because of the way the document root directory is specified for Mac OS web servers (see the discussion of Mac OS web servers, above). For Mac OS web servers and multi-hosting plug-ins that implement version 1.3 of the W*API, ServletExec will use the configuration options for the default server for unconfigured servers.

For single-host environments on Windows, UNIX, and Mac OS, do not configure a virtual server. Instead, use (or modify) the configuration options for the default server.

Configuring a Virtual Server

Figure 11 on page 28 illustrates how to configure a virtual server using the ServletExec Admin UI. In order to configure a virtual server you must specify the host name of the server (i.e., “www.abc.com”) in the Server Name field and the path to the Servlets Direc-

tory. For Windows and UNIX web servers the Servlets Directory path must be specified as a full path. For Mac OS web servers the Servlets Directory path may be specified as either a full path or a relative path from the web server's root directory (in the latter case, ServletExec will display the directory as a full path after you click "Submit"). Multiple virtual servers may share a Servlets Directory.

The screenshot shows a web form for configuring a virtual server. It has five text input fields and two buttons. The fields are labeled: 'Server Name' (containing 'default'), 'Servlets Directory' (containing 'C:\inetPub\ServletExec ISAPI\Servlets\'), 'User Name' (empty), 'Allowed IPs' (containing '*. *.*.*'), and 'Error Page' (empty). The 'Submit' and 'Reset' buttons are located to the right of the 'Error Page' field.

Server Name:	default
Servlets Directory:	C:\inetPub\ServletExec ISAPI\Servlets\
User Name:	
Allowed IPs:	*.*.*
Error Page:	

Submit Reset

Figure 11. The “default” Virtual Server

For Windows, use the “\” character as the file separator when specifying directory paths; for UNIX, use “/”; for Mac OS, use “:”. All path specifications must end with the file separator; if you omit it, ServletExec will add it for you. Here are some examples of path specifications for the Servlets directory:

Windows:	C:\InetPub\ServletExec ISAPI\Servlets\
UNIX:	/usr/netscape/suitespot/docs/
Mac OS (relative):	:ServletExec 2.2:Servlets:
Mac OS (full):	Power HD:WebSTAR:ServletExec 2.2:Servlets:

For Mac OS web servers you must also specify the path to the document Root Directory for the virtual server. This directory must be specified as a relative path from the web server's root directory and must match the document root directory in the configuration of the virtual server plug-in (see the discussion of Mac OS web servers, above). If the web servers' root directory is to be used as the virtual server's root directory, you must specify “:” for the Root Directory entry. Here are some examples:

:	specifies that the virtual server's document root directory is the same as the web server's root directory
:domain1:	specifies that the virtual server's document root directory is the “domain1” sub-directory within the web server's root directory

The Error Page field is used to specify a URL that will be invoked by ServletExec when an error occurs during processing of a servlet for the virtual server. URLs must be specified as relative paths from the web server root, for example:

```
/error.html
/servlet/MyErrorServlet
```

The User Name and Allowed IPs fields are discussed below.

Administering Virtual Servers

After configuring virtual servers, any time you access a ServletExec Admin UI page, the host name of the virtual server is displayed at the top of the page. “default” is displayed for the default server (prior to configuring virtual servers all pages operate on the default server, and “default” is omitted).

The user name and password defined for a virtual server provide access to the configuration options for that server. For Microsoft IIS on Windows NT, the “Password” field is not displayed. Instead, the “User Name” must be that of a Windows NT User. The password defined for the Windows NT User is used for authenticating access to the ServletExec Admin UI.

The Allowed IPs field allows you to specify a comma-separated list of IP addresses from which clients are allowed to access the ServletExec Admin UI for a virtual server. The specified IP addresses may include the “*” character to represent all IP addresses in a subrange. For example, the following list of IP addresses would allow access to the ServletExec Admin UI from a client with the IP address 168.121.97.133, or any IP address in the range from 204.84.126.1 to 204.84.126.255:

`168.121.97.133, 204.84.126.*`

Access to the ServletExec Admin UI is always allowed from the local machine on which ServletExec is running regardless of the setting of the Allowed IPs field.

To access the admin pages for a virtual server, use the following URL:

`http://<server name>/servlet/admin`

where <server name> is the host name of the virtual server. If the user name and password for a virtual server are not defined (are left blank), only the ServletExec Admin user (as defined on the Register page) can access the configuration options for that server. The ServletExec Admin user name and password can be used to access the options for any virtual server.

To access the configuration options for the default server, use the host name of an unconfigured server or the IP address of the server as the <server name> in the admin URL. If multiple hardware virtual servers are defined, use the IP address of any virtual server. Only the ServletExec Admin user is allowed to access the configuration options for the default server.

Only the ServletExec Admin user is allowed to access the following ServletExec Admin UI pages:

- Register
- Logging
- Virtual Servers

- Security
- VM Settings (or Classpath)
- Session Tracking

Compatibility with Older Browsers

In order to support multiple virtual servers, ServletExec relies on receiving the server host name from the browser in the Host field of the HTTP request headers. Some older browsers do not set the Host field; when receiving requests from these browsers, ServletExec uses the configuration for the default server. If there is no default server configured, the browser will receive a “404 Not Found” result. If there is a default server configured, servlets that access files (such as the internal SSIServlet) may have problems because they may not receive the proper document root directory for the virtual server.

The following browsers are known to support the HTTP request header Host field:

- Netscape Navigator 2.0 and higher (Windows, UNIX, and Mac OS)
- Microsoft Internet Explorer 3.0 and higher (Windows)
- Microsoft Internet Explorer 2.1 and higher (Mac OS)

Security

By default, servlets that reside on the local web server have very broad security privileges and almost unlimited access to services provided by the host operating system. Remote servlets, on the other hand, are always untrusted and severely limited security privileges (see the section on configuring servlets, above).

It’s possible to limit the security privileges of local servlets, and to define such limits per virtual server. This feature is particularly useful to web hosting service providers who are hosting multiple domains on a single physical server, because servlet security privileges and restrictions can be set per domain.

The following security privileges can be enabled or disabled for local servlets per virtual server:

- Networking (Listen), when enabled, allows servlets to create network sockets to listen for incoming connections.
- Networking (Connect), when enabled, allows servlets to create network sockets to initiate outgoing connections.
- System Properties (Read), when enabled, allows servlets to read Java VM system properties. This privilege should remain enabled in most cases because some system classes (such as PrintWriter) need access to system properties.
- System Properties (Write), when enabled, allows servlets to write Java VM system properties.
- Start New Process, when enabled, allows servlets to invoke external (native) operating system processes.

- Load Dynamic Library, when enabled, allows servlets to load native libraries. This privilege must remain enabled if servlets use third party libraries that invoke native modules (such as the JDBC-ODBC bridge).
- Create Class Loader, when enabled, allows servlets to create custom Java class loaders. This privilege should remain enabled in most cases because the ServletExec built-in SSIServlet, JSPServlet, and JIServlet must be able to create custom class loaders.
- Set Factory, when enabled, allows servlets to set socket and stream handler factories.
- Print Job Access, when enabled, allows servlets to initiate print job requests.
- Clipboard Access, when enabled, allows servlets to access the system clipboard.

In addition to specifying these privileges, it's possible to define the directories and files to which servlets have read and/or write access. The value "all" gives servlets unlimited access to directories and files (this is the default).

If you decide to restrict servlet access to specific directories, keep in mind that servlets *must* have access to the following directories:

1. the Servlets directory for the virtual server
2. the properties directory for the virtual server, for example:
`C:\InetPub\ServletExec ISAPI\ServletExec Data\<virtual server>`
3. the virtual server's document root directory
4. all of the paths in the ServletExec classpath, including the JDK classpath; normally this means giving access to the JDK installation directory (for example: C:\jdk1.1.6) and all directories added to the ServletExec classpath via the VM Settings page (see the next section)

VM Settings & Classpath

Figure 12 on page 32 illustrates the Java Virtual Machine (VM) settings that may be modified for ServletExec for Microsoft IIS and Netscape/NSAPI, and the classpath setting.

For Mac OS web servers, only the classpath setting may be modified. For ServletExec for Netscape/WAI and Apache, VM settings (including the classpath) are specified when the ServletExecWAI or ServletExecApache application is started (see Appendix B3, C1 and C2 for more details).

Selecting a VM

ServletExec for Microsoft IIS and Netscape/NSAPI support up to four "flavors" of VMs:

- Sun's Classic VM for JDK 1.1.x and 1.2.x
- Sun's HotSpot Performance Engine for JDK 1.2.x
- IBM's Developer Kit for Windows, Java Technology Edition, Version 1.1.x

- Microsoft VM

ServletExec will use Sun's Classic VM by default. To use one of the other VMs, select the "Sun HotSpot", "IBM", or "Microsoft" setting. If one of these settings is selected, ServletExec will look for the selected VM during initialization. If the selected VM isn't installed, ServletExec will attempt to initialize using the Sun Classic VM.

The information at the top of the VM Settings page shows which VM ServletExec selected during initialization, as illustrated in Figure 13. Note that you must stop and restart your web server for changes to the selected VM to take effect.

Changes on this page take affect only after a server restart.

Native Stack Size:	128	K Bytes
Java Stack Size:	400	K Bytes
Minimum Heap Size:	1024	K Bytes
Maximum Heap Size:	16384	K Bytes
Verbose GC:	☐ Enabled ☒ Disabled	
Verbose:	☐ Enabled ☒ Disabled	
JITC:	☒ Enabled ☐ Disabled	
Class GC:	☒ Enabled ☐ Disabled	
Async GC:	☒ Enabled ☐ Disabled	
Verify:	☐ None ☒ Remote ☐ All	
Java VM:	☒ Sun Classic ☐ Sun HotSpot ☐ IBM ☐ Microsoft	

Enter additional directories to add to the Java VM classpath, one per input field.

C:\inetPub\ServletExec ISAPI\classes
C:\Java\Archives\MyArchive.jar
Submit Reset

Figure 12. Java VM Settings & Classpath

Other VM Settings

Under normal circumstances, modifying the VM settings (other than selecting the Java VM and enabling/disabling the JITC) is unnecessary, potentially dangerous, and not recommended.

Java VM: Classic VM 1.2.1 from Sun Microsystems Inc.
VM Info: build JDK-1.2.1-A, native threads, symcjit
JITC: symcjit
Host: Windows NT 4.0 on x86

Figure 13. Java VM Information

The just-in-time-compiler (JITC) setting only affects JDK 1.1.6 and greater, and is enabled by default. Enabling the JITC with older versions of the JDK has no effect. Disabling the JITC may help resolve compatibility problems with third-party libraries that contain native code.

Additional notes for VM Settings:

- The Native Stack Size, Java Stack Size, Async GC, and Verify settings are not available with JDK 1.2.
- The JDK 1.2 Solaris Production release requires a minimum of 2048K for the Minimum Heap Size. ServletExec enforces this minimum by automatically using a value of 2048K if the configured value is less than that.

Classpath

The Java VM classpath defines the directories in which the VM will search when trying to load class files. By default, ServletExec sets the classpath to include all directories required by ServletExec.

Additional directories can be added to the classpath via the ServletExec Admin UI. This is required if you have a common set of class files (third-party libraries, for example) that are shared by many servlets and must be placed outside of the Servlets directory. Or, it may be useful in the presence of multiple virtual servers where each virtual server has its own Servlets directory.

For Windows, use the “\” character as the file separator when specifying directory paths; for UNIX, use “/”; for Mac OS, use “:”. Directories that are added to the classpath must be specified as full paths.

For Windows, paths to network drives must use the full UNC name and not a mapped drive letter. UNC names take the form:

\\<machine name>\<drive share name>\<path to file>

This is because mapped drive letters are associated with particular users. ServletExec runs as part of the web server, which runs as an NT service and therefore is not aware of network drive letter mappings.

You must stop and restart the web server before changes to the VM settings or classpath take effect. (Microsoft IIS and PWS users please refer to Appendix A for instructions on stopping and restarting your web server).

Session Tracking

ServletExec's session tracking feature can be configured via the Admin UI. See Chapter 4 for a further discussion of the session tracking feature. The following configuration options may be set as illustrated in Figure 14 on page 35:

Session Tracking - enables/disables Session Tracking. The default value is enabled.

URL Rewriting - enables/disables the rewriting of URLs to place the session ID in the URLs for session tracking. The default value is disabled.

Protocol Switch Rewriting - enables/disables the rewriting of URLs to place the session ID in the URLs when the URL indicates a switch from "http" to "https" or vice-versa. The default value is disabled.

Cookies - enables/disables the use of cookies for the session ID. The default value is enabled.

Persistence - enables/disables restoring of sessions after a server restart. The default value is enabled.

Swap Directory - specifies the sub-directory within the server directory (within the ServletExec Data directory) where sessions are stored when they are swapped out. The default value is sessionSwap.

Swap Interval - the interval specified in seconds when ServletExec will check for too many sessions in memory. The default value is 10 seconds.

Max Residents - the maximum number of sessions stored in memory at one time. After this limit is reached, sessions will be swapped to disk. The default value is 1024.

Invalidation Interval - the interval specified in seconds when ServletExec will check for sessions that have become invalid. The default value is 10 seconds.

Invalidation Time - the time specified in minutes that a session can go unused before becoming invalid. The default value is 30 minutes.

Session Tracking: ☒ Enabled ☐ Disabled

URL Rewriting: ☐ Enabled ☒ Disabled

Protocol Switch Rewriting: ☐ Enabled ☒ Disabled

Cookies: ☒ Enabled ☐ Disabled

Persistence: ☒ Enabled ☐ Disabled

Max Residents:

Swap Directory:

Swap Interval: Seconds

Invalidation Interval: Seconds

Invalidation Time: Minutes

Figure 14. Session Tracking Settings

If cookies are enabled for Session Tracking then the following settings are also supported, as illustrated in Figure 15:

Cookie Name - the name used for the session ID cookie. The default value is “sesessionid”.

Cookie Comment - the value of the comment field that is sent with the session ID cookie. The comment field is only sent to clients that support RFC2109 cookies. If Cookie Comment is left blank then a comment field isn't sent. The default value is “ServletExec Session Tracking Cookie”.

Cookie Domain - the value of the domain field that is sent with the session ID cookie. If Cookie Domain is left blank then a domain field isn't sent. The default value is null (i.e., a blank field).

Cookie Max Age - the maximum age of the session ID cookie. If Cookie Max Age is set to a value less than 0 then the expires field isn't sent for Netscape cookies and the Max-Age field isn't sent for RFC2109 cookies. The default value is -1.

Cookie Path - the value of the path field that is sent with the session ID cookie. If Cookie Path is left blank then a path field isn't sent. The default value is “/”.

Cookie Secure - if true, the secure field is sent with the session ID cookie. The default value is false.

Cookie Name:	sesessionid
Cookie Comment:	ServletExec Session Tracking Cookie
Cookie Domain:	
Cookie Max Age:	-1
Cookie Path:	/
Cookie Secure:	☐ True ☒ False

Figure 15. Session Tracking Cookie Settings

More information about cookies can be found in RFC2109 and in Netscape's Cookie Specification, which can be retrieved from:

http://home.netscape.com/products/newsref/std/cookie_spec.html

IIS Security

Because of the way ServletExec filters HTTP requests in order to forward them to servlets, the built-in security of the Microsoft IIS server cannot be used to protect URLs that invoke servlets. To work around this limitation, ServletExec provides a way to protect such URLs. On the IIS Security/Groups admin page, as illustrated in Figure 16 on page 36, ServletExec Groups can be defined that contain Windows NT Users.

IMPORTANT! In order for this feature to work properly, you **must** disable HTTP Keep-Alives for your web site. For IIS 4.0, open the Properties dialog for your web site and select the Performance tab. Uncheck the "HTTP Keep-Alives Enabled" checkbox.

On the IIS Security/Resources admin page, access to a servlet or URL that invokes a servlet can be restricted to specified ServletExec Groups and/or Windows NT Users, as illustrated in Figure 17.

ServletExec Group	Windows NT Users
adminGroup	Administrator, paul, vince
timeGroup	gregg, ken

Figure 16. IIS Security Groups

To protect a servlet enter the servlet name as the resource and then enter the ServletExec Groups and/or Windows NT Users that are allowed to access to the servlet. To protect a URL that invokes a servlet enter the URL as the resource and then enter the groups and/or Windows NT Users who have access to the servlet. Entering a URL resource will cause all URLs that begin with the URL resource to be protected.

Resource	ServletExec Groups	Windows NT Users
/private	adminGroup	
DateServlet	timeGroup	keeper, watcher

Figure 17. IIS Security Resources

Conclusion

The ServletExec Admin UI provides you with full access to ServletExec's rich feature set in an easy-to-use, browser-based interface. This chapter described the use of the ServletExec Admin UI in detail. The remaining chapters of this document describe the advanced features of ServletExec.

Server-Side Includes (SSI)

Creating Dynamic Pages with ServletExec

ServletExec implements a suite of server-side includes (SSI) functions via its built-in SSI Servlet. These functions allow you to embed the output of servlets into SSI pages, implement page counters, include the contents of other files in an SSI page, conditionally show and hide portions of an SSI page, and more.

SSI Overview

The internal Server-Side Includes (SSI) Servlet supports the <SERVLET> tag as specified for the Java Web Server 1.1. The Java Web Server documentation for the <SERVLET> tag can be retrieved from:

http://jserv.javasoft.com/products/java-server/documentation/webserver1.1/servlets/core_servlets.html

In addition to the <SERVLET> tag, the SSI Servlet supports SSI commands based on the NCSA definition. The NCSA SSI specification can be retrieved from:

<http://hoohoo.ncsa.uiuc.edu/docs/tutorials/includes.html>

The <SERVLET> tag can be used to insert the output of a Servlet into the response page. The SSI commands can be used to insert information about the incoming request, conditionally display sections of HTML and include the contents of a separate file.

Netscape Enterprise SSI

The Netscape Enterprise Server (NES) has a native SSI implementation that supports many of the features of the ServletExec SSI Servlet. ServletExec allows you to invoke servlets from an NES SSI page using the include SSI command. For example:

```
<!--#include virtual="/jsp10test1.jsp"-->  
<!--#include virtual="/servlet/DateServlet"-->
```

For additional information on NES server-side includes, refer to the NES documentation.

WebSTAR SSI

WebSTAR includes its own SSI plug-in that supports many of the features of the ServletExec SSI Servlet. ServletExec allows you to invoke servlets from a WebSTAR SSI page via two methods:

1. For WebSTAR 2.1 and greater, the `piservice` command can be used in a WebSTAR SSI page to have ServletExec invoke a servlet. The format of this command is:

```
<!--#exec piservice="ServletExec" param="url"-->
```

The `url` parameter is the URL you would normally use to invoke the servlet, without the protocol and host fields. For example, to invoke the `TestServlet` with query arguments the following command would be used:

```
<!--#exec piservice="ServletExec"
      param="/servlet/TestServlet?name=value"-->
```

Finally, the WebSTAR SSI plug-in decodes `%20` in URLs to spaces which causes problems for ServletExec. Therefore, you should encode spaces in URLs as plus signs (+). For example:

```
<!--#exec piservice="ServletExec"
      param="/servlet/TestServlet?company=New+Atlanta"-->
```

2. For WebSTAR 3.0 and greater, the `<SERVLET>` tag can be used in a WebSTAR SSI page to have ServletExec invoke a servlet. The format of this tag is the same as for ServletExec's SSI Servlet as described below. (Be sure to remove the WebSTAR Servlet Runner from the Plug-Ins folder when using ServletExec).

For additional information on the WebSTAR SSI plug-in refer to the WebSTAR manual.

Configuring SSI

ServletExec's SSI features are implemented by an internal servlet. This servlet is implemented by the class `newatlanta.servletexec.SSIServlet` and by default is configured with the Servlet Name "SSIServlet" (see the section on configuring servlets in Chapter 2). The SSIServlet accepts a single optional initialization argument:

`errorPage` – specifies a URL to be invoked when a JSP error occurs (for example: `/error.html` or `/servlet/ErrorServlet`).

Also by default, a suffix alias of `*.shtml` is defined for the SSIServlet (see the section on configuring servlet aliases in Chapter 2). This means that the SSIServlet will process any URL request for a file ending with the extension `".shtml"`.

Some web servers also use the `".shtml"` extension for their own built-in SSI processing. For these servers, in order to use both the SSIServlet and the web server's SSI features,

you must either reconfigure the web server or modify the SSIServlet alias configuration to something other than “*.shtml” (for example, “*.ssi”).

ServletExec ships with a sample SSI page named “ssiexample.shtml”. Place this file in your web server’s document root directory, then invoke the SSIServlet via the following URL:

```
http://www.yourserver.com/ssiexample.shtml
```

Replace “www.yourserver.com” in the above URL with the host name or IP address of your web server.

The SSIServlet may be used in servlet chains and as a response filter.

<SERVLET> Tag

The <SERVLET> tag has the following format:

```
<SERVLET NAME="servlet name" CODE="servlet class" CODEBASE="servlet
    codebase" init1="value1" init2="value2">
<PARAM NAME="param name" VALUE="param value">
.
.
</SERVLET>
```

The NAME parameter of the SERVLET tag corresponds to the servlet name as configured using the ServletExec Admin UI (see the section on configuring servlets in Chapter 2). If the NAME matches a configured servlet name, then the remaining parameters of the SERVLET tag (CODE, CODEBASE, and initialization arguments) are ignored. The name parameter value is case-sensitive.

If the NAME parameter is provided and does not match a configured servlet name, then the remaining parameters of the SERVLET tag are used to load and initialize the servlet. The servlet remains loaded and subsequent requests using the same NAME are processed by the servlet (that is, the servlet is not reloaded or reinitialized for subsequent requests).

If the NAME parameter is not provided, then the remaining parameters of the SERVLET tag are used to load and initialize the servlet. After processing the request, the servlet is destroyed (that is, the servlet is reloaded and reinitialized for every request).

The CODE parameter of the SERVLET tag specifies the name of the class that implements the doGet(), doPost(), or service() methods of the servlet. The CODE parameter value is case-sensitive and must include full package information for the class (for example: newatlanta.demo.TestServlet). The CODE parameter is required if the NAME parameter does not match a configured servlet, or if the NAME parameter is not

provided. The CODE parameter is ignored if the NAME parameter matches a configured servlet.

The CODEBASE parameter of the SERVLET tag specifies the location of class files for remote servlets. The CODEBASE parameter is not required for servlets whose class files reside on the local web server. The CODEBASE parameter is ignored if the NAME parameter matches a configured servlet.

The initialization arguments (from the example: `init1="value1" init2="value2"`) are passed to the servlet's `init()` method as `name=value` pairs. The initialization arguments are ignored if the NAME parameter matches a configured servlet.

The PARAM tag is used to specify parameters that are passed to the servlet every time it is invoked to process a request. The values for the NAME and VALUE parameters are used to create `name=value` pairs that are merged with the search and POST arguments from the HTTP request and passed to the servlet. The values for the name and value parameters can be placed inside single or double quotes.

SSI Commands

SSI commands have the following format:

```
<!--#command tag1="value1" tag2="value2" ... tagN="valueN" -->
```

The commands and tags are case-insensitive, and the values are case-insensitive for all commands except for the config command with the `errmsg` and `timefmt` tags.

The list of commands is:

echo	insert the value of a variable into the response page
include	insert the contents of another file into the response page
fsize	insert the size of a file into the response page
config	configure SSI for this page
show	conditionally display a section of HTML
hide	conditionally hide a section of HTML
counter	increment and optionally display counters
random	generate a random number for the page
nossi	stop SSI processing for this page

Each of the SSI commands is described in detail below.

Echo

The echo command is used to insert the value of a variable into the response page. For example, to insert the current document name the following syntax would be used:

```
<!--#echo var="document_name" -->
```

The variables that can be displayed are listed below:

document_name	the current document name (i.e., filename.shtml)
document_uri	the virtual path to this document
query_string_unescaped	unescaped version of search query string
query_string	escaped version of search query string
date_local	local date and time
date_gmt	local date and time in Greenwich mean time
server_software	name and version of server software
server_name	server's host name, DNS alias or IP address
server_protocol	name and revision of protocol used by request
server_port	the port to which the request was sent
request_method	the request method (i.e., GET)
path_info	extra path information
path_translated	virtual-to-physical translation of path info
remote_host	remote's host name or its IP address if host name isn't known
remote_addr	remote's IP address
auth_type	the authentication type
remote_user	the user name sent to be authenticated
content_type	content type of data for POST and PUT requests
content_length	content length of data for POST and PUT requests
http_xxx	any HTTP header value (i.e., http_user_agent)
random	the current random value for the response page

In addition, the echo command can be used to display the value of a counter or the date of the last time the counter was zeroed. Here's an example of how to display this information:

```
<!--#echo count="counter_name" -->
<!--#echo lastzero="counter_name" -->
```

If the `count` or `lastzero` tag is used on a counter that doesn't exist then a new counter is created with a value of 0 and a `lastzero` value of the current date and time. Refer to the section on the `count` command to learn more about counters.

Multiple variables and counters can be used with one `echo` command. In this case a comma is used to separate the displayed values. For example:

```
<!--#echo var="document_name" count="daycnt" -->
```

Include

The `include` command is used to include a separate file into the response page. It can even be used to include a separate file which contains server-side includes. In this case the file must have an extension which matches the suffix alias for the `SSIServlet`. A file can be included by specifying either its virtual path or its path relative to the current document.

For example, either of the following 2 statements could be used to include `header.html` into `default.html` where both pages are in the directory `Test`:

```
<!--#include virtual="/Test/header.html" -->
```

or

```
<!--#include file="header.html" -->
```

A virtual path must begin with a `/` while a file path can not begin with a `/`.

An included file is treated as part of the original file. Therefore any SSI commands in an included file will continue to take affect in the original file. For example, the results of a `config`, `hide` or `show` command will continue to take affect in the original file. The one exception to this is the `nossi` command.

Multiple files can be specified with one `include` command. For example:

```
<!--#include virtual="/Test/header1.html" file="header2.html" -->
```

Fsize

The `fsiz` command is used to insert the size of a file into the response page. A file can be specified using the `virtual` and `file` tags as described for the `include` command. For example:

```
<!--#fsiz file="header.html" -->
```

Multiple files can be specified with one `fsiz` command. In this case a comma is used to separate the inserted file sizes.

Config

The `config` command is used to configure SSI processing for a page. The items that can be configured are:

errmsg defines an error message to be used instead of the built-in error messages. For example:

```
<!--#config errmsg="Customized error message." -->
```

timefmt specifies the format to be used to display a date and time. The value must be a standard UNIX time format string. The default time format is "%Y/%m/%d:%H:%M:%S". For example:

```
<!--#config timefmt="%A, %B %#d, %Y %H:%M:%S" -->
```

sizefmt specifies the format to be used to display a file size. The possible values are abbrev and bytes (the default). The abbrev value causes the file size to be output in kilobytes while the bytes value causes the file size to be output in bytes. For example:

```
<!--#config sizefmt="abbrev" -->
```

Multiple items can be configured with one config command. For example:

```
<!--#config errmsg="Customized error message." sizefmt="abbrev" -->
```

Show

The show command is used to conditionally show a block of HTML. The syntax for the show command is as follows:

```
<!--#show var1="testvalue1" var2="testvalue2" ... varN="testvalueN" -->
```

All of the variables listed under the echo command can be used in a show command. If more than 1 test is specified then the show command uses AND logic to determine if the HTML should be shown. If one of the tests fails then the following HTML is not shown until a show command is processed which equates to true. The show command can be used with no tests to force the displaying of HTML. The test values can have operators specified. The allowable operators are:

```
"testvalue" - equals
"=testvalue" - equals
"!testvalue" - does not equal
"testvalue*" - begins with
"*testvalue" - ends with
"*testvalue*" - contains
">testvalue" - greater than
">=testvalue" - greater than or equal
"=>testvalue" - greater than or equal
"<testvalue" - less than
"<=testvalue" - less than or equal
"= <testvalue" - less than or equal
```

The test values are treated as strings when performing the test except when the variables `random`, `server_port` and `content_length` are involved. In these cases the test values are treated as integers. The tests are case-insensitive.

Here's an example of only displaying a block of HTML for Internet Explorer:

```
<!--#show http_user_agent="*MSIE*" -->
Only show this HTML to Internet Explorer.
<!--#show -->
```

Here's an example of randomly displaying 4 different headers:

```
<!--#show random="<=25" -->
Header 1.
<!--#show random=">25" random="<=50" -->
Header 2.
<!--#show random=">50" random="<=75" -->
Header 3.
<!--#show random=">75" random="<=100" -->
Header 4.
<!--#show -->
```

Hide

The `hide` command is used to conditionally hide a block of HTML. The syntax for the `hide` command is as follows:

```
<!--#hide var1="testvalue1" var2="testvalue2" ... varN="testvalueN" -->
```

The `hide` command is very similar to the `show` command. When all of the tests in a `hide` command equate to true, the following HTML is hidden until a `show` command is processed which equates to true.

Here's an example of hiding a block of HTML from Internet Explorer:

```
<!--#hide http_user_agent="*MSIE*" -->
Hide this HTML from Internet Explorer.
<!--#show -->
```

Counter

The `counter` command is used to increment and optionally display a counter. To only display a counter the `echo` command should be used. The valid tags for the `counter` command are `incr` and `echo`. The tag `incr` causes the counter to be incremented while the tag `echo` causes the counter to be incremented and displayed. The syntax for the `counter` command is:

```
<!--#counter incr="counter_name" -->
```

or

```
<!--#counter echo="counter_name" -->
```

A generic counter can have any name as long as it doesn't conflict with one of the special counters listed below:

daycnt	a daily hit counter for the page
weekcnt	a weekly hit counter for the page
monthcnt	a monthly hit counter for the page
totalcnt	a total hit counter for the page

If the counter command is used on a counter which doesn't exist then the counter is created and incremented to 1.

Counters can be zeroed or removed by going to the SSI Servlet admin page at the following URL:

```
http://your.hostname.com/servlet/admin
```

See the section on configuring Server-Side Includes in Chapter 2.

Counters are loaded when the SSI Servlet is initialized and written out when the SSI Servlet is destroyed.

Random

The random command is used to generate a new random number for a page. A random number is automatically generated for a page when the SSI Servlet receives the request. In some cases it may be useful to generate a new random number in the middle of parsing a SSI page. For example, this could be used to generate a random header and footer which use their own unique random number. The syntax for the random command is:

```
<!--#random -->
```

Nossi

The nossi command is used to indicate that the file contains no more SSI commands and that the SSI Servlet can stop parsing the file. This can be used to speed up the parsing of a file. The syntax for the nossi command is:

```
<!--#nossi -->
```

Session Tracking

Tracking Visitors to Your Web Site

ServletExec implements the full Session Tracking feature of the Java Servlet API as defined by JSDK 2.1. This feature allows you to maintain state information about a user visiting your web site. Administration of the Session Tracking feature in ServletExec has been designed to closely match that of the Java Web Server 1.1. Documentation on the Session Tracking feature as implemented by the Java Web Server 1.1 can be retrieved from:

http://jserv.javasoft.com/products/java-server/documentation/webserver1.1/session_track/SessionTr.html

Session Tracking Summary

Session Tracking gives servlets the ability to keep state about a user as the user moves through the web site. ServletExec maintains user state by creating an HttpSession object for each visitor to the site. The HttpSession object is then passed as part of the request to the servlet's `doGet()`, `doPost()`, and `service()` methods. Servlets can add information to the HttpSession objects or read information from them.

When a servlet receives a request, it invokes the `getSession()` method of the `HttpServletRequest` object to obtain an existing HttpSession object or create a new one. After obtaining an HttpSession object, the servlet can invoke its `putValue()` method to add state information to the object and its `getValue()` method to get state information from the object. "State information" can be any arbitrary named objects defined by the servlet developer.

ServletExec tracks the HttpSession object for a user via an internal session ID. If cookies are enabled via the ServletExec Admin UI and the client supports cookies then cookies are used to keep track of the session ID. If the client supports RFC2109 cookies then RFC2109 cookies will be used otherwise Netscape cookies will be used. More information about cookies can be found in RFC2109 and in Netscape's Cookie Specification, which can be retrieved from:

http://home.netscape.com/products/newsref/std/cookie_spec.html

If cookies cannot be used and URL rewriting is enabled via the ServletExec Admin UI then URL rewriting will be used to keep track of the session ID. URL rewriting is disabled by default.

A session can be invalidated either manually by the servlet developer by invoking the `HttpSession` object's `invalidate()` method, or automatically by `ServletExec`. A session will be invalidated automatically when it has been unused for a time period specified by the "Invalidation Time" setting.

Session Tracking Configuration

The Session Tracking feature is configured via the `ServletExec` Admin UI. See Chapter 2 for a detailed discussion of the `ServletExec` Admin UI. The following configuration options are available:

Session Tracking - enables/disables Session Tracking. The default value is enabled.

URL Rewriting - enables/disables the rewriting of URLs to place the session ID in the URLs for session tracking. The default value is disabled.

Protocol Switch Rewriting - enables/disables the rewriting of URLs to place the session ID in the URLs when the URL indicates a switch from "http" to "https" or vice-versa. The default value is disabled.

Cookies - enables/disables the use of cookies for the session ID. The default value is enabled.

Persistence - enables/disables restoring of sessions after a server restart. The default value is enabled.

Swap Directory - specifies the sub-directory within the server directory (within the `ServletExec` Data directory) where sessions are stored when they are swapped out. The default value is `sessionSwap`.

Swap Interval - the interval specified in seconds when `ServletExec` will check for too many sessions in memory. The default value is 10 seconds.

Max Residents - the maximum number of sessions stored in memory at one time. After this limit is reached, sessions will be swapped to disk. The default value is 1024.

Invalidation Interval - the interval specified in seconds when `ServletExec` will check for sessions that have become invalid. The default value is 10 seconds.

Invalidation Time - the time specified in minutes that a session can go unused before becoming invalid. The default value is 30 minutes.

If cookies are enabled for Session Tracking then the following settings are also supported:

Cookie Name - the name used for the session ID cookie. The default value is "sesessionid".

Cookie Comment - the value of the comment field that is sent with the session ID cookie. The comment field is only sent to clients that support RFC2109 cookies. If

Cookie Comment is left blank then a comment field isn't sent. The default value is "ServletExec Session Tracking Cookie".

Cookie Domain - the value of the domain field that is sent with the session ID cookie. If Cookie Domain is left blank then a domain field isn't sent. The default value is null (i.e., a blank field).

Cookie Max Age - the maximum age of the session ID cookie. If Cookie Max Age is set to a value less than 0 then the expires field isn't sent for Netscape cookies and the Max-Age field isn't sent for RFC2109 cookies. The default value is -1.

Cookie Path - the value of the path field that is sent with the session ID cookie. If Cookie Path is left blank then a path field isn't sent. The default value is "/".

Cookie Secure - if true, the secure field is sent with the session ID cookie. The default value is false.

Session Tracking Example

Example 1 on page 50 illustrates use of the Session Tracking feature. This example tracks the number of time a visitor has accessed a particular page and displays it as a counter.

The first thing this servlet does is retrieve the HttpSession object for the user by invoking the HttpServletRequest getSession() method. Passing true as the parameter causes ServletExec to create a new HttpSession object if one doesn't already exist.

The servlet then attempts to retrieve an Integer object named "counter" from the HttpSession object by invoking the getValue() method. If getValue() returns null, the servlet creates a new Integer object, initializing it to 1. Otherwise, the servlet increments the value of the integer object.

The Integer object is then placed back into the HttpSession by invoking putValue() so it can be retrieved again when the next request from this user is processed.

The servlet then outputs an HTML page back to the user including the incremented Integer value.

Finally, if the Integer value exceeds 100, the HttpSession is invalidated. This will cause a new HttpSession object to be created for this user the next time this servlet is accessed.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SessionTest extends HttpServlet
{
    public void doGet( HttpServletRequest request,
                      HttpServletResponse response )
        throws ServletException, IOException
    {
        // get the session object (create a new one if necessary)
        HttpSession session = request.getSession( true );

        // get the counter from the session object and increment it
        Integer ival = (Integer)session.getValue( "counter" );

        ival = new Integer( ival == null ? 1 : ival.intValue() + 1 );

        // write out the new session data value
        session.putValue( "counter", ival );

        // output the page
        response.setContentType( "text/html" );

        ServletOutputStream out = response.getOutputStream();

        out.println( "<html><head><title>Session Tracking " );
        out.println( "Test</title></head>" );
        out.println( "<body><center><h1>Session Tracking Test</h1>" );
        out.println( "You have hit this page " + ival + " times." );
        out.println( "</center></body></html>" );
        out.close();

        // invalidate the session if over 100 hits
        if ( ival.intValue() > 100 )
            session.invalidate();
    }
}
```

Example 1. Session Tracking

HttpSession Value Classes

The example above stores and retrieves objects of class `Integer` via the `putValue()` and `getValue()` methods of the `HttpSession` object. The `Integer` class is part of the base `java.lang` package. Servlet developers are free to store and retrieve objects of any class as values in the `HttpSession` object, however, the class files for such classes must not be placed in the `Servlets` directory. Instead, they must be placed in a different directory that must then be added to the `ServletExec` classpath (see Chapter 2 for discussions of the `Servlets` directory and the `ServletExec` classpath).



Presentation Templates

Defining a Common Look for Your Web Site

ServletExec implements the Presentation Templates feature defined by the Java Web Server 1.1. This feature allows you to define a common look for HTML pages on your web site. Documentation on the Presentation Templates feature as implemented by the Java Web Server 1.1 can be retrieved from:

<http://jserv.javasoft.com/products/java-server/documentation/webserver1.1/templates/templates.html>

The Presentation Templates feature is implemented by ServletExec's built-in Template Servlet, which matches the implementation of the Java Web Server 1.1.

Presentation Templates Summary

The Presentation Templates feature allows you to apply common elements as well as common content to a set of HTML pages without actually changing the pages themselves. The common elements are contained in a single file, called a template file, which is given the name `default.template`.

The `default.template` file is placed in the topmost directory for the set of HTML pages that will have the template applied. Each directory can have its own `default.template` file. If a directory doesn't have a `default.template` file then the Template Servlet will keep moving up the directory tree until it finds a `default.template` file.

The topmost directory for the set of pages which will have templates applied can be mapped to the Template Servlet using a servlet prefix alias. Alternately, the Template Servlet can be mapped to a set of HTML pages using a servlet suffix alias. See the section heading Template Servlet Configuration, below.

Presentation Templates may also be applied to the output of servlets by using the Template Servlet in servlet chains or as a response filter. See the section heading Using Presentation Templates with Servlets, below.

Template Servlet Configuration

In order to use the Template Servlet, you must configure the servlet class `newatlanta.servletexec.TemplateServlet` with the servlet name `TemplateServlet`. You must then configure servlet aliases that map to the `TemplateServlet` for the HTML files that will have templates applied to them (see Chapter 2 for instructions on how to configure servlet names and aliases).

The Template Servlet may be mapped to either prefix or suffix aliases. In either case, the Template Servlet uses the path portion of the request URL as the directory in which to begin searching for the `default.template` file. The Template Servlet may be mapped to multiple aliases (as any servlet may).

To test your setup, copy the `template` and `images_template` directories from the `ServletExec Examples` directory to the web server's root directory and create a servlet alias for `/template` that maps to the `TemplateServlet`. Then enter the following URL in a browser:

```
http://<hostname>/template/index.html
```

The contents of the `template` directory will be served via the Template Servlet.

Using Presentation Templates with HTML Pages

For a set of HTML files, a `default.template` file is created which defines the common look for the HTML files. The `default.template` file must contain at least the HTML tag, a HEAD section, a BODY section and a closing HTML tag. It should also contain the following tags:

```
<subst data="HEAD"/> or <subst data="HEAD"></subst>
```

This tag indicates where the `TemplateServlet` should place the HTML from the source document's HEAD section.

```
<subst data="BODY"/> or <subst data="BODY"></subst>
```

This tag indicates where the `TemplateServlet` should place the HTML from the source document's BODY section.

In addition to these tags, the `default.template` file can also contain the following tag:

```
<subst data="<element>"/> or <subst data="<element>"></subst>
```

This tag is used to insert HTML that has been defined as an element in the `default.definitions` file. For example, an element called "Trademark" could be assigned a value in the `default.definitions` file as shown below:

```
Trademark=<B>All trademarks herein are the property of their  
respective owners.</B>
```

This HTML could then be inserted in the template file by simply using the tag

```
<subst data="Trademark"/>
```

The `default.template` file needs to be placed in the topmost directory for the set of HTML files that will have the template applied. Each directory can have its own `default.template` file. If a directory doesn't have a `default.template` file then the `TemplateServlet` will keep moving up the directory tree until it finds a `default.template` file.

When the `Template Servlet` is invoked via a prefix alias, templates are only applied to those files which have a mime type of `text/html`. For performance reasons, files that don't have a mime type of `text/html` shouldn't be placed in a template directory.

When an element is used in a `default.template` file, the `TemplateServlet` will first look at the `default.definitions` file in the same directory for a definition. If it doesn't find a definition for the element then it will keep moving up the directory tree until it finds a definition in a `default.definitions` file in a parent directory.

Using Presentation Templates with Servlets

The `Template Servlet` can be used in servlet chains or as a response filter in order to apply Presentation Templates to the output of servlets. (See Chapter 2 for instructions on how to configure servlet chains and response filters).

By definition, a servlet chain must be invoked via a servlet alias. When used in a servlet chain, the `Template Servlet` uses the path portion of the request URL as the directory in which to begin searching for the `default.template` file.

The `Template Servlet` can be configured as a response filter for a MIME type such as `template/html` (for example). In this case, the output of any servlet that sets the MIME type to `template/html` (via the `HttpServletResponse.setContentType()` method) will be processed by the `Template Servlet`. The `Template Servlet` uses the path portion of the request URL as the directory in which to begin searching for the `default.template` file, therefore, the original servlet must be invoked via a servlet alias.



JavaServer™ Pages

Server-side Scripting using Java

ServletExec 2.2 implements the final JavaServer Pages (JSP) 1.0 specification; for backwards compatibility with previous releases, ServletExec 2.2 also supports JSP Draft Specification 0.91. *New development should conform to the 1.0 specification; the 0.91 draft specification is supported only for backwards compatibility with pages developed for previous versions of ServletExec.*

More information about JSP, including a copy of the JSP 1.0 specification, can be found on Sun Microsystems' web site:

<http://java.sun.com/products/jsp/>

ServletExec's built-in JSP10Servlet implements the JSP 1.0 specification; the built-in JSPServlet, as in previous versions of ServletExec, implements Draft Specification 0.91. The remainder of this chapter only discusses the JSP10Servlet.

JavaServer Pages Summary

The JavaServer Pages feature involves generating HTML pages from hybrid HTML/Java source files. These source files are typically named using the `.jsp` extension (though this isn't a requirement). Java source code is embedded in `.jsp` pages using JSP tags that are described in the JSP 1.0 specification.

The JSP10Servlet performs the following tasks the first time a `.jsp` file is requested:

1. Translate the `.jsp` file into a Java source file (`.java`).
2. Compile the Java source file into a Java class file (`.class`).
3. Instantiate and run the compiled Java class file as a servlet.

For subsequent requests for the `.jsp` page, if the requested file was modified after its initial compilation, the JSP10Servlet detects the modification and repeats the above tasks. If the requested file was not modified, the JSP10Servlet uses the existing servlet to handle the page request.

By default, the `javac` compiler included with the JDK is used to compile the Java source file in step 2. The JSP10Servlet can be configured to use an external (non-JDK) compiler

such as IBM's `jikes` or Microsoft's `jvc` (see discussion of the `JSP10Servlet` compiler parameter, below). If the compiler encounters errors while compiling a `.jsp` page, the output error messages are sent to the client's browser.

JavaServer Pages Configuration

In order to use the JSP feature, the following configuration steps must be performed:

1. Configure the `JSP10Servlet`.
2. Assign a servlet alias to the `JSP10Servlet`.
3. Add `tools.jar` to the `ServletExec` classpath (JDK 1.2 only).

The rest of this section discusses each of these configuration steps and concludes with instructions for testing your configuration.

Configure the `JSP10Servlet`

In order to use the `JSP10Servlet`, the servlet class:

```
newatlanta.servletexec.JSP10Servlet
```

must be configured via the `ServletExec` Admin UI with the servlet name `JSP10Servlet`. See Chapter 2 of the User Guide for instructions on how to configure servlet names. *The `JSP10Servlet` is preconfigured in `ServletExec 2.2` and does not need to be modified unless you need to change the value of the `init` arguments, discussed below.*

Optional initialization (`init`) arguments may be assigned when the `JSP10Servlet` is configured. None of the `init` arguments are preconfigured and all take their default values unless explicitly modified via the `ServletExec` Admin UI. The `JSP10Servlet` supports the following `init` arguments:

`verbose` – if true, prints a message to `System.out` when compiling a file. The default is false.

`defaultPageClassName` – indicates the default base class for JSP pages. The default base class can be overridden by using the `extends` JSP directive. The default value for this argument is `JSP10HttpJspPage`.

`compileCommand` – specifies the compiler options to use when compiling the Java files (for example: `compileCommand=-classpath C:\MyClasses`). The default value for this argument is `null`.

`compiler` – specifies the path to the executable for an external (non-JDK) Java compiler to be used to compile the JSP pages (on Windows the path to the executable must not include the `.exe` extension). `ServletExec 2.2` has been tested with IBM's `jikes` and Microsoft's `jvc` compilers; if one of these compilers is used, the `compileCommand` argument **must** be used to specify the classpath (see

above). The default value for this argument is `null`, which causes the JDK `javac` compiler to be used.

`compileAfterRestart` – if `true`, then after a web server restart all `.jsp` pages are recompiled even if they haven't been modified. This is necessary so that `.class` files used by a `.jsp` page which may have been modified will be recompiled (even if the `.jsp` page hasn't changed). Normally this situation occurs during development, so this parameter should be set to `true` during development and reset to `false` for production. The default value is `false`.

`workingDir` – the directory for storing the generated `.java` and `.class` files. The default value for this argument is the `Servlets` directory (see Chapter 2 for a discussion of the `Servlets` directory).

`packagePrefix` – the package to prepend to the class name. The default value for this argument is `pagecompile`.

`packageLevel` – specifies the number of sub-directories from the root document directory path to be prepended to the class name as packages. This argument only needs to be used when virtual servers are used, but not configured via `ServletExec` Admin UI. In this case, the value of `1` for this argument should be adequate to insure that the JSP pages generated for each virtual server have a unique class name. The default value is `0`.

`pageCheckSeconds` – the time in seconds to wait before checking if a JSP page has changed. The default value for this argument is `0`.

`errorPage` – specifies a URL to be invoked when a JSP error occurs (for example: `/error.html` or `/servlet/ErrorServlet`).

`urlRewriting` – if `true`, then all URLs in a JSP page are rewritten with the Session ID if URL rewriting is enabled for Session Tracking. The default value is `false`.

Assign a Servlet Alias

In order to use the `JSP10Servlet`, the `JSP10Servlet` name must be mapped to the servlet suffix alias `*.jsp`. See Chapter 2 for instructions on configuring servlet names and aliases. *The `JSP10Servlet` name and `*.jsp` alias are preconfigured in `ServletExec 2.2` and do not need to be modified unless you want to map JSP pages using a different suffix alias.*

Add `tools.jar` to the Classpath (JDK 1.2)

If you're using JDK 1.2 and the default `javac` compiler for compiling JSP pages, the file `tools.jar` must appear in the `ServletExec` classpath. If you're running Microsoft IIS or Netscape Enterprise on Windows, `ServletExec 2.2` will automatically add `tools.jar` to its classpath and you don't need to do anything. If you're running any other web server,

you may need to add `tools.jar` to the ServletExec classpath or start script. See Chapter 2 and the appropriate appendix for your web server for instructions on adding directories to the ServletExec classpath.

Testing Your Configuration

To test your JSP configuration, do the following:

1. Place the file `hangman.jsp` in the web server's document root directory (`hangman.jsp` can be found in the Examples subdirectory of the ServletExec installation directory).
2. Add the bean `newatlanta.bean.HangmanBean` to the ServletExec classpath. For example, the full path to the `Hangmanbean.class` file for the default Microsoft IIS installation is:

```
C:\InetPub\Scripts\ServletExec ISAPI\Examples\newatlanta\bean\Hangmanbean.class
```

Therefore, the following path should be added to the ServletExec classpath:

```
C:\InetPub\Scripts\ServletExec ISAPI\Examples
```

See Chapter 2 for discussion of the ServletExec classpath. Be sure to stop and restart your web server after modifying the ServletExec classpath.

3. Invoke the `hangman.jsp` page from a browser using the following URL:
`http://<server-name>/hangman.jsp`

If you have problems running this test, check the `ServletExec.log` file for error messages.

Using JavaServer Pages

The `JSPServlet` reads a `.jsp` file that contains embedded JSP directives, JSP declarations, JSP scriptlets, JSP expressions and JSP beans, parses the file and creates a servlet that generates the HTML response page. Refer to the JSP 1.0 specification for a complete description of the JSP syntax:

<http://java.sun.com/products/jsp/download.html>

<SERVLET> Tag

`.jsp` pages may include the `<SERVLET>` tag in the same format as defined for SSI pages in Chapter 3.

Invoking a JSP page from a Servlet

A servlet can invoke a JSP page using the `RequestDispatcher` interface introduced in JSDK 2.1. Here's an example:

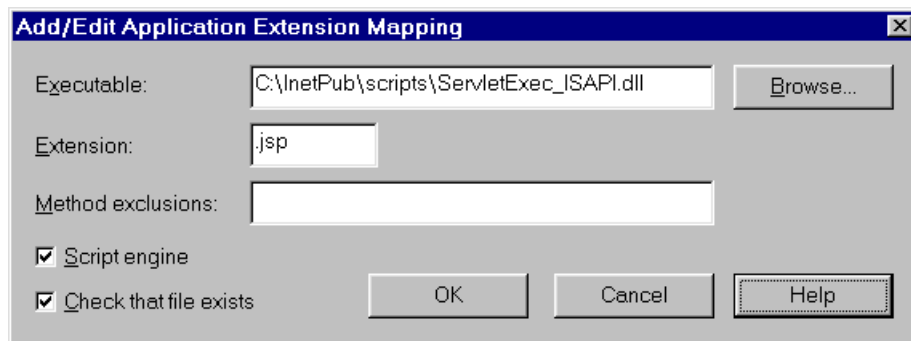
```
RequestDispatcher dispatcher =  
    getServletContext().getRequestDispatcher( "/mypage.jsp" );
```

```
dispatcher.include( request, response );  
// request and response are the parameters to the servlet's service(),  
// doGet(), or doPost() method
```

Microsoft IIS Extension Mapping

The Extension Mapping feature of Microsoft IIS 4.0 can be used to control access to JSP pages via NT File System (NTFS) security. Perform the following steps to enable this feature:

1. Stop the IIS Admin Service using the Services control panel.
2. Edit the `servers.properties` file (which can be found in the ServletExec Data folder) and set the value of the `servlethexec.useiisextmapping` property to `true`.
3. Add `.jsp` to the Microsoft IIS extension map:
 - a. Open the Internet Service Manager application (Microsoft Management Console).
 - b. Open the Properties dialog for your server. First, expand the Internet Information Server entry until you can see the icon for your server. Then right-click the server icon and select Properties from the pop-up menu.
 - c. Make sure "WWW Service" is selected in the Master Properties combo-box, then click the Edit button.
 - d. In the WWW Service Master Properties dialog, select the Home Directory tab, then click the Configuration button. This will open the Application Configuration dialog.
 - e. In the Application Configuration dialog, click the Add button to open the Add/Edit Application Extension Mapping dialog.
 - f. In the Add/Edit Application Extension Mapping dialog, set the Executable field to the path to the `ServletExec_ISAPI.dll` file; set the Extension field to `.jsp`; and, make sure the "Check that file exists" checkbox is checked. The dialog should appear as follows:



Click OK to close all of the dialogs.

IMPORTANT! After setting the value of the `servlethexec.useiisextmapping` property to `true` in step 2, above, you **must** configure all ServletExec suffix aliases using IIS Ex-

tension Mapping. If the suffix alias does not map to a physical file (unlike .jsp) then do not check the “Check that file exists” checkbox in the Add/Edit Application Extension Mapping dialog for that suffix alias.



Page Compilation

Embedding Java in HTML Pages

ServletExec implements the Page Compilation feature defined by the Java Web Server 1.1. This feature allows you to embed Java in HTML pages. Documentation on the Page Compilation feature as implemented by the Java Web Server 1.1 can be retrieved from:

http://jserv.javasoft.com/products/java-server/documentation/webserver1.1/page_comp/intro.html

The Page Compilation feature is implemented by ServletExec's built-in JIServlet (Java Invoker Servlet), which matches the implementation of the Java Web Server 1.1.

Page Compilation is a “first generation” technology that is being replaced by JavaServer Pages (JSP). ServletExec 2.2 supports both Page Compilation and JSP (see Chapter 6, above).

Page Compilation Summary

Page compilation is the process of generating HTML pages from hybrid HTML/Java source files. Java is embedded in the hybrid HTML/Java pages using `<java>...</java>` tags. The JIServlet performs the following tasks the first time a hybrid HTML/Java file is requested:

1. Translate the HTML/Java file into a Java source file (. java).
2. Compile the Java source file into a Java class file (. class).
3. Instantiate and run the compiled Java class file as a servlet.

For subsequent requests for the hybrid HTML/Java page, if the requested file was modified after its initial compilation, the JIServlet detects this and repeats these tasks. If the requested file was not modified, the JIServlet uses the existing servlet to handle the page request.

The javac compiler is used to compile the Java source file in step 2. If the javac compiler encounters errors while compiling a hybrid HTML/Java page, the output error messages are sent to the client's browser.

Page Compilation Configuration

In order to use the Page Compilation feature, the following configuration steps must be performed:

1. Configure the JIServlet.
2. Assign a servlet alias to the JIServlet.
3. Add `tools.jar` to the ServletExec classpath (JDK 1.2 only).

The rest of this section discusses each of these steps and concludes with instructions for testing your configuration.

Configure the JIServlet

In order to use the JIServlet, the servlet class `newatlanta.servletexec.JIServlet` must be configured with the servlet name JIServlet. See Chapter 2 for instructions on how to configure servlet names. The JIServlet name is preconfigured in ServletExec 2.2.

Optional initialization (`init`) arguments may be assigned when the JIServlet name is configured. The JIServlet supports the following `init` arguments:

`verbose` - If `true`, prints a message when compiling a file. The default is `false`.

`defaultPageClassName` - Indicates the default base class for JI pages. The default base class can be overridden by using the `<java type=extends>` tag. The default value for this argument is `HttpServlet`.

`compileCommand` - the compiler options to use when compiling the Java files (for example: `compileCommand=-deprecation`). The default value for this argument is `null`.

`compileAterRestart` - if `true`, then after a web server restart all `.jhtml` pages are recompiled even if they haven't been modified. This necessary so that `.class` files used by a `.jhtml` page which may have been modified will be recompiled (even if the `.jhtml` page hasn't changed). Normally this situation occurs during development, so this parameter should be set to `true` during development and reset to `false` for production. The default value is `false`.

`workingDir` - the directory for storing the generated `.java` and `.class` files. The default value for this argument is the Servlets directory (see Chapter 2 for a discussion of the Servlets directory).

`packagePrefix` - the package to prepend to the class name. The default value for this argument is `pagecompile`.

Assign a Servlet Alias

In order to use the JIServlet, the JIServlet name must be mapped to the servlet suffix alias “*.jhtml”. See Chapter 2 for instructions on how to configure servlet names and aliases. *The JIServlet name and *.jhtml alias are preconfigured in ServletExec 2.2.*

Add tools.jar to the Classpath (JDK 1.2)

If you’re using JDK 1.2 and the default javac compiler for compiling JSP pages, the file tools.jar must appear in the ServletExec classpath. If you’re running Microsoft IIS or Netscape Enterprise on Windows, ServletExec 2.2 will automatically add tools.jar to its classpath and you don’t need to do anything. If you’re running any other web server, you may need to add tools.jar to the ServletExec classpath or start script. See Chapter 2 and the appropriate appendix for your web server for instructions on adding directories to the ServletExec classpath.

Testing Your Configuration

To test your setup, place the file test.jhtml (from the ServletExec Examples folder) in the web server’s root directory and invoke it from a browser using the following URL:

`http://<server-name>/test.jhtml`

If you have problems running this test, check the ServletExec.log file for error messages.

Using Page Compilation

The JIServlet reads an HTML file with embedded <java>...</java> tags (a .jhtml file), parses the file and creates a servlet that generates the HTML response page. The generated .java and .class files are placed in the same directory as the .jhtml file. Whenever the .jhtml file is modified, the servlet is re-created.

The tags supported by the JIServlet are:

<java></java> - defines Java code for the service() method. Any Java code within this tag is compiled into the servlet’s service() method. The following variables are defined in the service() method and can be used within these tags:

request - an instance of the HttpServletRequest class, this is the HTTP request object passed as a parameter to the service() method

response - an instance of the HttpServletResponse class, this is the HTTP response object passed as a parameter to the service() method.

out - an instance of the ServletOutputStream class, this is the object retrieved from the response.getOutputStream() method.

<java type=code></java> - same as <java></java>.

<java type=import></java> - encloses the names of classes to be imported.

`<java type=extends></java>` - encloses the name of a class from which to extend the servlet. By default, the servlet extends `HttpServlet` or the class specified by the `defaultPageClassName` initialization (init) argument (see the heading *Configure the JIServlet*, above).

`<java type=implements></java>` - encloses the names of interfaces that the servlet implements.

`<java type=class></java>` - encloses member variables and methods.

`<java type=print></java>` - encloses a Java expression to be sent to the output stream.

back quote operator - encloses a Java expression inside a tag to be sent to the output stream. Functionally similar to the `<java type=print>` tag, this can be used to include Java expressions within HTML tags.

syntax: `<tag 'java expression'>`

example: `<IMG SRC="`fileNameVariable`">`

`<servlet>` - .jhtml pages may include the `<servlet>` tag in the same format as defined for SSI pages in Chapter 3.

Page Compilation Example

Example 2 on page 64 illustrates the use of embedded Java in .jhtml pages. This example keeps a counter of the number of times the servlet has been accessed and produces an HTML page displaying the value of the counter.

Known Limitations

1. `ServletExec 2.2` does not support the `defaultEncoding` initialization argument, which is supported by the `Java Web Server 1.1`.
2. Since the `defaultEncoding` initialization argument is not supported, the `out` variable in the `service()` method will always be of type `ServletOutputStream`.

```
<java type=import>
java.net.*
</java>

<java type=class>
    // The class tag can be used to define instance variables and
    // methods for the class. In this example, we are defining the
    // instance variable counter and the method init.

    int counter = 0;    // the number of times this servlet instance
                        // has been invoked

    // NOTE: be sure to call super.init( config ) when overriding the
    //        HttpServlet init method.
    public void init( ServletConfig config ) throws ServletException
    {
        System.out.println( "Invoked init() for test.jhtml." );
        super.init( config );
    }
</java>

<html>
<head>
<title>JIServlet Test</title>
</head>
<body>
<center>

<h1>JIServlet Test</h1>

<p>This servlet has been invoked

<java type=code>
    // The code tag is used to place Java code in the service() method
    counter++;
</java>

<java type=print>
counter
</java>

times.</p>

</center>
</body>
</html>

<java type=class>
    public void destroy()
    {
        System.out.println( "Invoked destroy() for test.jhtml." );
        super.destroy();
    }
</java>
```

Example 2. Page Compilation

File Upload Servlet

Uploading Files to Your Web Server

ServletExec implements a File Upload feature which allows you to upload files to your web server using the protocol specified in RFC1867. Using this feature requires a browser that supports the RFC1867 protocol, such as Netscape Communicator 4.0. The File Upload feature is implemented by ServletExec's built-in Upload Servlet.

Upload Servlet Configuration

In order to use the Upload Servlet, you must configure the servlet class `newatlanta.servletexec.UploadServlet` and give it the servlet name `UploadServlet` (see the Chapter 2 for instructions on how to configure servlet names). The Upload Servlet is pre-configured in ServletExec 2.2.

To test your setup, place the file `upload.html` (from the ServletExec Examples directory) in the web server's root directory and invoke it from a browser.

The following `init` arguments are supported by the Upload Servlet (servlet `init` arguments can be configured via the ServletExec Admin UI; see Chapter 2 for instructions):

`defaultDir` – the default upload directory. The default value is the directory named “upload” in the web server's root document directory.

`defaultSuccessPage` – the default page to return when the upload is completed successfully. The default value is `null` which indicates the Upload Servlet's built-in success page should be used.

`defaultErrorPage` – the default page to return when the upload fails. The default value is `null` which indicates the Upload Servlet's built-in error page should be used.

`debug` – if `true`, debug messages are displayed as the files are uploaded. The default value is `false`. On Windows, these messages are sent to the `ServletExec.log` file and to the DBMON console if DBMON is being used (see Appendix E for a discussion of DBMON). On the Mac OS, these messages are sent to the web server's console.

`limit` – the value of this parameter takes the form `n<K|k|M|m>` (for example: `limit=2M` or `limit=500k`) and can be used to specify the maximum length of an uploaded file. If this limit is exceeded, an error message is logged and the browser will report a broken connection. By default, there is no limit to the size of uploaded files.

The `defaultDir`, `defaultSuccessPage` and `defaultErrorPage` can be specified as either full paths or relative paths from the web server's root document directory. For Windows and UNIX, use `“/”` as the path separator; for Mac OS, use `“:”`. Relative paths must begin with the path separator while full paths do not.

Upload Feature Summary

The Upload Servlet supports the uploading of multiple files at the same time. Name/Value pairs can be sent to the Upload Servlet to modify its behavior for a specific upload. The Name/Value pairs that are supported are:

`uploadDir` – this name/value pair can be used before each file that is being uploaded to indicate to which directory it should be uploaded.

`successPage` – this name/value pair can be used to indicate the page which should be returned if the upload completes successfully. The value for the `successPage` can be specified as either a full path or a relative path from the web server's root document directory. For Windows and UNIX, use `“/”` as the path separator; for Mac OS, use `“:”`. Relative paths must begin with the path separator while full paths do not.

`errorPage` – this name/value pair can be used to indicate the page which should be returned if the upload fails. The value for the `errorPage` can be specified as either a full path or a relative path from the web server's root document directory. For Windows and UNIX, use `“/”` as the path separator; for Mac OS, use `“:”`. Relative paths must begin with the path separator while full paths do not.

`uploadFileName` – this name/value pair can be used to specify the name of the uploaded file.

Servlet Chaining

The Upload Servlet can be used in servlet chains as long as it is the first servlet in the chain (see Chapter 2 for a discussion of servlet chaining). Servlets in the chain after the Upload Servlet can retrieve the request parameters using the `HttpServletRequest` methods `getParameterNames()`, `getParameterValue()`, and `getParameterValues()`. `ServletExec` adds an additional request parameter named `success` that contains a boolean value indicating success or failure of the Upload Servlet.

Known Limitations

1. Uploads have been tested and work properly with the following browsers:
 - Netscape Navigator 3.0 (Windows and Mac OS)
 - Netscape Communication 4.0 (Windows and Mac OS)
 - Microsoft Internet Explorer 4.0 (Windows)
2. Uploads have been tested and do not work with the following browsers:
 - Microsoft Internet Explorer 3.0 (Windows and Mac OS)
 - Microsoft Internet Explorer 4.0 (Mac OS)
3. The largest file that has been tested is 3M bytes.



Appendix A. Microsoft IIS/PWS Installation

Verifying Your ServletExec Installation

This appendix contains important information that will allow you to verify your installation of ServletExec ISAPI for Microsoft IIS/PWS. It will also be useful if you decide to uninstall ServletExec to make sure you've completely removed all installed components.

ServletExec for Microsoft IIS/PWS is implemented using Microsoft's ISAPI server programming interface and supports the standard Java Virtual Machine (VM) for Windows from Sun Microsystems, the IBM Developer Kit for Java (Windows Edition), and the Microsoft VM. ServletExec ISAPI runs as an in-process extension of your web server for maximum performance and reliability.

Upgrading from a Previous Version

In order to upgrade to a new version of ServletExec and maintain your old configuration settings, perform the following steps:

1. Stop your web server. For IIS 3.0, you must stop *all* services (WWW, FTP, and Gopher). For IIS 4.0, you must stop the IIS Admin Service from the Services control panel.
2. Make backup copies of the ServletExec Data and Servlets sub-directories of the ServletExec ISAPI directory. The default location for the ServletExec ISAPI directory is within the InetPub directory (or the WebShare directory for Personal Web Server 1.0 on Windows 95).
3. Uninstall the old version of ServletExec using the Add/Remove Programs control panel. The ServletExec ISAPI directory and the ServletExec Data, Servlets, and Servlet Logs sub-directories will not be deleted by the uninstaller. You may delete the Servlet Logs sub-directory, but leave the others alone.
4. Run the installer for the new version of ServletExec. Choose the existing ServletExec ISAPI directory when prompted for a "Destination Folder".
5. Restart your web server.

After restarting your web server, the new version of ServletExec should run using your old configuration settings. If you have any problems, you can restore the ServletExec Data and Servlets sub-directories from the backups you made in step 2, above.

System Requirements

- ServletExec 2.2 supports the following operating systems and web servers:

<u>Operating System</u>	<u>Web Server</u>
Windows NT 4.0 Server	Internet Information Server (IIS) 2.0, 3.0, and 4.0
Windows NT 4.0 Workstation	Peer Web Services 2.0 and 3.0, and Personal Web Server (PWS) 4.0
Windows 95/98	Personal Web Server (PWS) 1.0 and 4.0

- In order to install ServletExec 2.2 you must first install either the Java Development Kit (JDK) or the Java Runtime Environment (JRE) version 1.1.x or 1.2 from Sun Microsystems, the IBM JDK or JRE for Windows version 1.1.x, or the Microsoft VM. If you plan to use the Microsoft VM we recommend that you first upgrade to the latest version.

The Sun JDK or JRE 1.1.x for Windows can be downloaded from:

<http://java.sun.com/products/jdk/1.1/index.html>

<http://java.sun.com/products/jdk/1.1/jre/index.html>

The Sun JDK or JRE 1.2 for Windows can be downloaded from:

<http://java.sun.com/products/jdk/1.2/index.html>

<http://java.sun.com/products/jdk/1.2/jre/index.html>

The IBM JDK or JRE 1.1.x for Windows can be downloaded from:

<http://www.ibm.com/developer/java/>

The latest version of the Microsoft VM can be downloaded from:

<http://www.microsoft.com/java/>

At the time of this writing, the latest production versions of the Sun JDKs for Windows are 1.1.8 and 1.2.2; the latest production version of the IBM JDK is 1.1.8; the latest production version of the Microsoft VM is build 3186. We recommend using these versions with ServletExec 2.2.

WARNING! Sun JDK 1.1.5 contains a memory leak, which makes it unsuitable for use with ServletExec in production environments. *Do not use JDK 1.1.5 with ServletExec!*

The JDK or JRE must be installed on a local drive and not on a mapped network drive. If the JDK is installed on a mapped network drive, ServletExec will not be able to load and initialize the Java VM.

Uninstall Other Servlet Engines

VERY IMPORTANT! You **must** uninstall any other servlet engines you may have previously installed for use with Microsoft IIS/PWS before installing and using ServletExec. In particular, the Filter DLL registry entries or ISAPI Filters metabase entries associated with other servlet engines must be removed (see the heading Registry & Metabase Entries, below).

What Was Installed?

When you installed ServletExec, three changes were made to your system:

- the ServletExec ISAPI directory was created
- the ServletExec_ISAPI.dll file was installed
- Registry and/or metabase entries for ServletExec were created or modified

The following sections describe each of these changes.

The ServletExec ISAPI Directory

The ServletExec ISAPI directory was created in the location you selected during the installation process. The default location suggested by the installer is within the InetPub directory (or the WebShare directory for Personal Web Server 1.0 on Windows 95); however, there are no restrictions on the location of this directory.

IMPORTANT! If you're using the NT File System (NTFS), permissions for the ServletExec ISAPI directory and its sub-directories *must* be set so that ServletExec has read and write access to these directories. Because ServletExec runs as part of the IIS process, it will run as different users at different times. The following Groups should be granted Full Control: SYSTEM and Authenticated Users. The following User should be granted Full Control: IUSR_<server name> (the user created by IIS for processing requests for anonymous users).

The ServletExec ISAPI directory contains the following files:

- DBMON debugging program (see Appendix E)
- JSDK License Agreement
- ServletExec License Agreement
- READ ME
- Release Notes

- ServletExec Debugging Tips
- ServletExec Admin shortcut

In addition to the files listed above, the ServletExec ISAPI directory contains the following sub-directories:

classes	this directory is automatically added to the ServletExec classpath; servlet or non-servlet classes can be placed in this directory; see the READ ME for more information
Documentation	contains this manual, a tutorial for writing servlets, and the Java Servlet API documentation from the Java Servlet Development Kit (JSDK) 2.1
Examples	contains examples of some of ServletExec's advanced features, including: Server-Side Includes (SSI), Page Compilation, JavaServer Pages (JSP), File Upload servlet, and Presentation Templates
lib	contains the servlet.jar and ServletExec22.jar files; these archives are automatically added to the ServletExec classpath, and are required for ServletExec operation; DO NOT MOVE OR DELETE THIS DIRECTORY OR THESE FILES
Servlet Logs	contains the Servlet.log files; entries are added to the Servlet.log file whenever a servlet invokes its log() method; see the section in Chapter 2 on configuring the servlet logging feature
ServletExec Data	contains ServletExec configuration files; DO NOT MODIFY THE CONTENTS OF THIS DIRECTORY; in emergency situations you may delete this entire directory and ServletExec will recreate it using its default configuration; however, if you do this all changes you've made to the default configuration will be lost and you will be required to re-enter your ServletExec serial number
Servlets	the default location for servlet class files; initially contains sample servlets that ship with ServletExec

Do not move the ServletExec ISAPI directory after installation. There is a registry entry that allows ServletExec to find this directory (see the discussion of registry entries, below); if you move it then ServletExec won't be able to find its configuration files.

ServletExec_ISAPI.dll

The ServletExec_ISAPI.dll dynamic link library (DLL) was installed in the directory you selected during the installation process. This directory *must* be mapped to a Microsoft IIS/PWS virtual directory and the virtual directory *must* have execute permission

enabled. The default location suggested by the installer is C:\InetPub\Scripts (or C:\WebShare\Scripts for Personal Web Server 1.0 on Windows 95), which is mapped to the SCRIPTS virtual directory.

For IIS 4.0, open the Properties dialog for the virtual directory that maps to the physical directory in which ServletExec_ISAPI.dll resides (by default, this is the SCRIPTS virtual directory). Verify that the Name parameter under Application Settings is not grayed out. If it is, click Create, Apply, and then OK so that it is not grayed out.

Registry & Metabase Entries

The ServletExec installer creates a new registry entry with the following key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\New Atlanta Communications\ServletExec ISAPI
```

This key contains a single parameter, Home, that contains the path to the ServletExec ISAPI directory. If you move the ServletExec ISAPI directory after installation, you must modify this key to contain the new path.

Filter DLLs Registry Entry

The ServletExec installer also modifies the Filter DLLs parameter for IIS 2.0 & 3.0 on Windows NT Server, Peer Web Services 2.0 & 3.0 on Windows NT Workstation, and PWS 1.0 & 4.0 on Windows 95/98. This parameter has the following key:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W3SVC\Parameters
```

The path to ServletExec_ISAPI.dll is added to the Filter DLLs parameter. **VERY IMPORTANT!** You *must* remove entries for other servlet engines you may have previously installed for Microsoft IIS from the Filter DLLs parameter. The uninstallers for most servlet engines do not automatically remove this entry.

Metabase ISAPI Filter Entry

The ServletExec 2.2 installer automatically modifies the IIS 4.0 (NT Server) or PWS 4.0 (NT Workstation) metabase to add the ISAPI Filter entry (the metabase is the “new and improved” registry). **VERY IMPORTANT!** You *must* remove entries for other servlet engines you may have previously installed. The uninstallers for most servlet engines do not automatically remove the metabase ISAPI Filter entry. Some servlet engines do not use the metabase ISAPI Filter entry for IIS/PWS 4.0, but instead continue to use the old Filter DLLs registry entry. Be sure to remove these entries as described above.

To examine or modify the metabase ISAPI Filter entry:

1. Open the Internet Service Manager application (Microsoft Management Console).
2. Open the Properties dialog for your server. First, expand the Internet Information Server entry until you can see the icon for your server. Then right-click the server icon and select Properties from the pop-up menu. You should see a dialog similar to Figure 18 on page A-6.

3. Make sure “WWW Service” is selected in the Master Properties combo-box (as illustrated in Figure 18), then click the Edit button.
4. Select the ISAPI Filters tab in the Master Properties dialog. The dialog should appear similar to Figure 19 on page A-7.
5. Use the Remove button to delete entries for other servlet engines you may have installed previously. Use the Edit button to examine or modify the ServletExec entry. Use the Add button to add the ServletExec entry if the installer did not add it successfully.

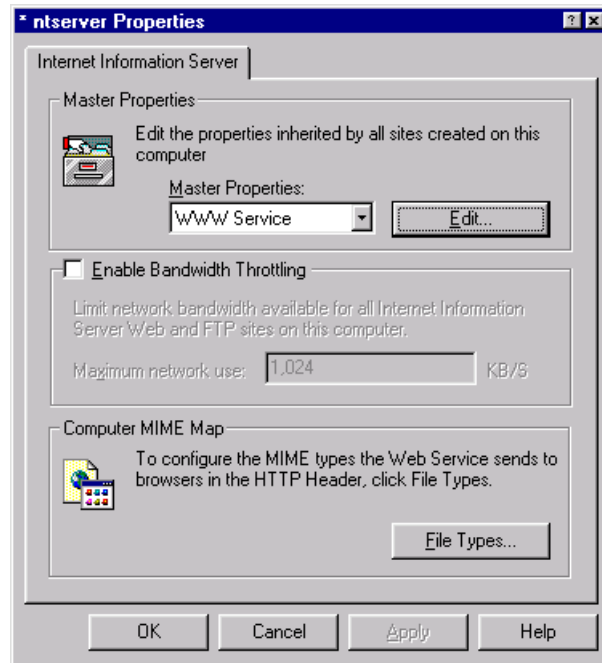


Figure 18. Server Properties Dialog

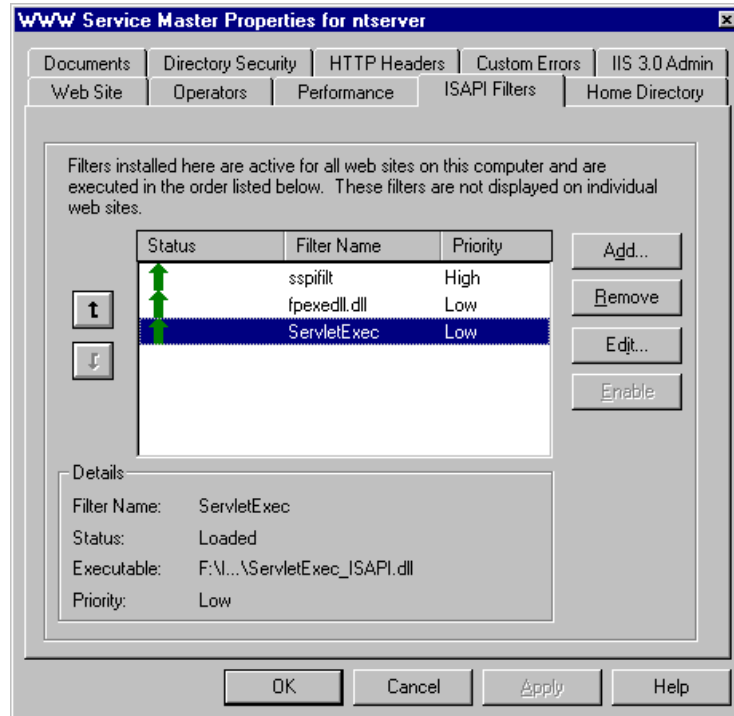


Figure 19. ISAPI Filters

ServletExec Admin User Name & Password

For Windows NT, the user name and password assigned in the ServletExec Admin User Interface must match those of a Windows NT user as defined in the User Manager. (See Chapter 2, heading Setting the Admin User Name & Password).

For IIS 4.0 (NT Server) and PWS 4.0 (NT Workstation), Basic Authentication must be enabled in order for the ServletExec Admin user name and password to work properly. Basic Authentication must be enabled in the WWW Service Master Properties and the individual web server Properties dialogs:

1. Open the Internet Service Manager (Microsoft Management Console).
2. Open the Properties dialog for your server. First, expand the Internet Information Server entry until you can see the icon for your server. Then right-click the server icon and select Properties from the pop-up menu. You should see a dialog similar to Figure 18 on page A-6.
3. Make sure “WWW Services” is selected in the Master Properties combo-box (as illustrated in Figure 18), then click the Edit button.
4. Click the Edit button.
5. Select the Directory Security tab.

- Under “Anonymous Access and Authentication Control,” click the Edit button. You should now see a dialog similar to Figure 20.

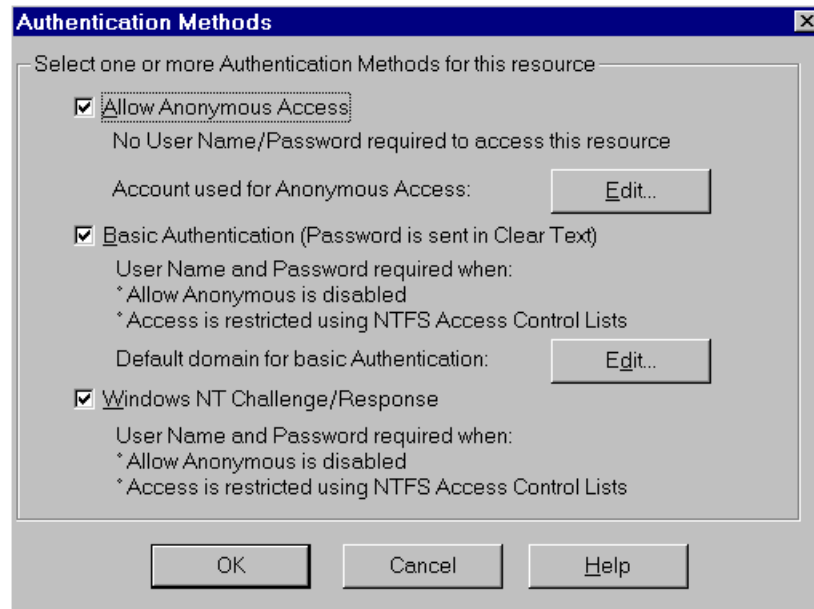


Figure 20. IIS Authentication Methods

- Make sure “Basic Authentication” is enabled.
- Click OK to close all dialogs.
- Open the Properties dialog for each configured web server (there may be only one, named “Default Web Server”).
- Repeat steps 6 through 9 for each configured web server.

JDK/JRE Installation

In order to install ServletExec 2.2 you must have first installed either JDK or JRE 1.1.x or 1.2. *The JDK or JRE must be installed on a local drive and not on a mapped network drive.* If the JDK is installed on a mapped network drive, ServletExec will not be able to load and initialize the Java VM.

It’s possible you may have multiple versions of the JDK installed on your system, or that you may install newer (or older) versions after installing ServletExec. ServletExec uses registry entries to determine which version of the JDK to use. It will look for an installed JDK first and if it doesn’t find one it will look for the JRE. Here’s the complete algorithm:

- Look for the following registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Development Kit

If found, go to step 2, otherwise look for the following key:

HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Runtime Environment

2. Read the CurrentVersion variable from the key found in step 1. Currently, the only valid values for this variable start with “1.1” or “1.2” (including, for example “1.1.7”).

3. Append the value of the CurrentVersion variable from step 2 to the key from step 1 to create a new key. For example:

HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Development Kit\1.1

or

HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Runtime Environment\1.1

4. Read the value of the JavaHome variable for the key from step 3 to find the location of the JDK or JRE.

If you launch DBMON and then restart your web server, ServletExec displays the Java VM settings in the DBMON console window during initialization (see Appendix E for a discussion of DBMON). You can examine the classpath displayed by ServletExec to see which version of the JDK is being used.

User Accounts for Microsoft IIS

Because ServletExec runs as part of the Microsoft IIS process, your servlets will run under different user accounts at different times. This will primarily affect their ability to read and write to the file system because access to the NT File System (NTFS) is based on the user account of the process. Here are the rules:

1. During normal request processing in your servlet’s `service()`, `doGet()`, or `doPost()` method, your servlet will be running under the user account of the authenticated user, if the user had to enter a username and password to access your servlet. Otherwise your servlet will be running under the IUSR_<server name> account (the account under which IIS runs anonymous user requests).
2. If your servlet is configured to be loaded by ServletExec during initialization, your `init()` method will be executed under the SYSTEM account. Otherwise, if your servlet is loaded when it receives its first request, the rules for item #1, above, apply.
3. Normally, your servlet’s `destroy()` method will run under the SYSTEM account when it is invoked due to a shutdown of ServletExec. However, if your servlet is reloaded due to a class file modification, the rules for item #1, above, apply.

Reinitializing ServletExec

In order to reinitialize ServletExec (which must be done after modifying the ServletExec classpath, for example), you must completely stop and restart IIS. This isn't as straightforward as you might think.

For IIS 2.0 & 3.0, you must stop *all* services (WWW, FTP, Gopher) to cause the ServletExec DLL to be unloaded. ServletExec will be reloaded and reinitialized when you restart the WWW service.

For IIS 4.0, in addition to stopping all services, you must stop the IIS Admin Service to cause the ServletExec DLL to be unloaded. This can be done from the Services control panel. Alternatively, you execute the batch file `stop_iis.bat`, which can be found in the ServletExec ISAPI directory, to completely stop IIS 4.0.

For PWS 4.0 on Windows 95/98, you must restart Windows in order to reload and reinitialize ServletExec.

Uninstalling ServletExec

Perform the following steps to completely uninstall ServletExec:

1. Stop your web server.
2. If you're running IIS 4.0, remove the ServletExec metabase filter entry as described under the heading Metabase ISAPI Filter Entry, above. If you're running any other version of IIS or PWS, use the Registry Editor to remove the ServletExec entry from the `Filter DLLs` parameter. This parameter has the following key:
`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W3SVC\Parameters`
3. If you're running PWS 4.0 on Windows 95/98, restart Windows.
4. Open the Control Panel and select Add/Remove Programs.
5. Select "ServletExec ISAPI 2.2" from the list and click Add/Remove.
6. The ServletExec uninstaller may not be able to remove all of the installed files. Remove the ServletExec ISAPI directory if it's still present. The default location for the ServletExec ISAPI directory is within the InetPub directory (or the WebShare directory for Personal Web Server 1.0 on Windows 95).



Appendix B1. Netscape/NSAPI Installation (Windows NT)

Verifying Your ServletExec Installation

This appendix contains important information that will allow you to verify your installation of ServletExec/NSAPI for Netscape FastTrack & Enterprise servers on Windows NT. It will also be useful if you decide to uninstall ServletExec to make sure you've completely removed all installed components.

In order to complete the ServletExec installation, the Netscape **obj.conf** configuration file must be modified. The ServletExec 2.2 installer gives you the option to allow it to automatically update the **obj.conf** file. Instructions in this appendix allow you to verify these changes, or to manually edit **obj.conf** if you chose not to let the installer do so automatically.

ServletExec for Netscape FastTrack & Enterprise servers on Windows NT is implemented using Netscape's NSAPI server programming interface, and the standard Java Virtual Machine (VM) for Windows from Sun Microsystems. This combination gives you maximum performance and Java compatibility. ServletExec NSAPI runs as an in-process extension of your web server.

Upgrading from a Previous Version

If you've installed ServletExec for multiple Netscape server instances, you only need to upgrade ServletExec for one instance and the upgrade will apply to all instances. In order to upgrade to a new version of ServletExec and maintain your old configuration settings, perform the following steps:

1. Stop your web server. If you've installed ServletExec for multiple Netscape server instances, stop them all.
2. Make backup copies of the ServletExec Data and Servlets directories for the Netscape server instance you're upgrading. The default location for these directories is within the C:\Netscape\SuiteSpot\Plugins\ServletExec NSAPI\https-<server name> directory.

3. Uninstall the old version of ServletExec for the Netscape server you're upgrading using the Add/Remove Programs control panel. The ServletExec NSAPI directory and its sub-directories will not be deleted by the uninstaller.
4. Delete the ServletExec_NSAPI.dll file from the ServletExec NSAPI directory. Delete all other files and sub-directories from the ServletExec NSAPI directory *except* for the https-<server name> sub-directories and their contents.
5. Run the installer for the new version of ServletExec. Choose the existing ServletExec NSAPI directory when prompted for a "Destination Folder".
6. Restart your web server.

After restarting your web server, the new version of ServletExec should run using your old configuration settings. If you have any problems, you can restore the ServletExec Data and Servlets sub-directories from the backups you made in step 2, above.

If you've installed ServletExec for multiple Netscape server instances, the Add/Remove Programs control panel will display the new ServletExec version number for the instance that you upgraded. All other server instances will display the old ServletExec version number. This is OK.

System Requirements

- ServletExec 2.2 for NSAPI only runs on Windows NT Server and Workstation version 4.0; it *does not* run on Windows 95/98.
- ServletExec 2.2 supports FastTrack & Enterprise servers version 3.0 and higher, including Enterprise 3.5, 3.5.1, and 3.6.
- In order to install ServletExec 2.2 you must first install either the Java Development Kit (JDK) or the Java Runtime Environment (JRE) version 1.1.x or 1.2 from Sun Microsystems, the IBM JDK or JRE for Windows version 1.1.x, or the Microsoft VM. If you plan to use the Microsoft VM we recommend that you first upgrade to the latest version.

The Sun JDK or JRE 1.1.x for Windows can be downloaded from:

<http://java.sun.com/products/jdk/1.1/index.html>

<http://java.sun.com/products/jdk/1.1/jre/index.html>

The Sun JDK or JRE 1.2 for Windows can be downloaded from:

<http://java.sun.com/products/jdk/1.2/index.html>

<http://java.sun.com/products/jdk/1.2/jre/index.html>

The IBM JDK or JRE 1.1.x for Windows can be downloaded from:

<http://www.ibm.com/developer/java/>

The latest version of the Microsoft VM can be downloaded from:

<http://www.microsoft.com/java/>

At the time of this writing, the latest production versions of the Sun JDKs for Windows are 1.1.8 and 1.2.2; the latest production version of the IBM JDK is 1.1.8; the latest production version of the Microsoft VM is build 3186. We recommend using these versions with ServletExec 2.2.

WARNING! Sun JDK 1.1.5 contains a memory leak, which makes it unsuitable for use with ServletExec in production environments. *Do not use JDK 1.1.5 with ServletExec!*

The JDK or JRE must be installed on a local drive and not on a mapped network drive. If the JDK is installed on a mapped network drive, ServletExec will not be able to load and initialize the Java VM.

Uninstall Other Servlet Engines

VERY IMPORTANT! You **must** uninstall any other servlet engines you may have previously installed for use with Netscape FastTrack/Enterprise servers before installing and using ServletExec. In particular, modifications you may have made to the **obj.conf** configuration file for other servlet engines must be removed.

Installing ServletExec for Multiple Servers

If you've created multiple server instances, you must run the ServletExec installer separately for each server for which you want to install ServletExec. Pay attention to the following items when installing ServletExec for multiple servers:

1. Stop all servers before running the ServletExec installer.
2. Use the same installation directory for all ServletExec installations.

The ServletExec installer will create a separate sub-directory within the ServletExec NSAPI directory for each server instance (see further discussion, below).

When installed for multiple Netscape server instances, each instance of ServletExec creates its own Java VM. Therefore, the servlets being hosted by ServletExec for each server are isolated from the others in separate VMs.

Note that you must purchase a unique ServletExec serial number for each Netscape server instance with which ServletExec will be installed.

Deactivate the Java Interpreter

For Enterprise 3.5.1 and 3.6, the built-in Java interpreter is deactivated by default; however, if it has been activated on your server you must deactivate it manually by performing the following steps:

1. From the Netscape Administration Server home page, select the server to be administered.
2. Select “Programs” from the menu bar in the upper frame of the server administration page.
3. Select the “Java” link from the menu in the left frame of the “Programs” page.
4. Turn off the Java interpreter.

This configuration step only applies to Enterprise 3.5.1 and 3.6. If you still have problems running ServletExec after deactivating the Java interpreter, check the `Init` directives in your **obj.conf** file as described below.

What Was Installed?

When you installed ServletExec, these changes were made to your system:

- the ServletExec NSAPI directory was created
- Registry entries for ServletExec were created or modified
- the Netscape **obj.conf** configuration file was updated (if you chose to let the ServletExec installer modify this file)

The following sections describe each of these changes.

The ServletExec NSAPI Directory

The ServletExec NSAPI directory was created in the location you selected during the installation process. The default location suggested by the installer is within the `C:\Netscape\SuiteSpot\Plugins` directory; however, there are no restrictions on the location of this directory.

The ServletExec NSAPI directory contains the following files:

- ServletExec_NSAPI.dll
- DBMON debugging program (see Appendix E)
- Install.log
- JSDK License Agreement
- ServletExec License Agreement
- READ ME

- Release Notes
- ServletExec Debugging Tips
- VerifyObjConf.class (see the discussion of VerifyObjConf.bat, below)

Within the ServletExec NSAPI directory are the following sub-directories:

Documentation	contains this manual, a tutorial for writing servlets, and the Java Servlet API documentation from the Java Servlet Development Kit (JSDK) 2.1
lib	contains the <code>servlet.jar</code> and <code>ServletExec22.jar</code> files; these archives are automatically added to the ServletExec classpath, and are required for ServletExec operation; DO NOT MOVE OR DELETE THIS DIRECTORY OR THESE FILES
Examples	contains examples of some of ServletExec's advanced features, including: Server-Side Includes (SSI), Page Compilation, JavaServer Pages (JSP), File Upload servlet, and Presentation Templates

Also within the ServletExec NSAPI directory are sub-directories for each server. Following Netscape conventions, the sub-directories are named `https-<server name>` (for Enterprise) or `httpd-<server-name>` (for FastTrack). Within each server sub-directory are the following directories:

classes	this directory is automatically added to the ServletExec classpath; servlet or non-servlet classes can be placed in this directory; see the READ ME for more information
Servlet Logs	contains the <code>Servlet.log</code> files; entries are added to the <code>Servlet.log</code> file whenever a servlet invokes its <code>log()</code> method; see the section in Chapter 2 on configuring the servlet logging feature
ServletExec Data	contains ServletExec configuration files; DO NOT MODIFY THE CONTENTS OF THIS DIRECTORY ; in emergency situations you may delete this entire directory and ServletExec will recreate it using its default configuration; however, if you do this all changes you've made to the default configuration will be lost and you will be required to re-enter your ServletExec serial number
Servlets	the default location for servlet class files; initially contains sample servlets that ship with ServletExec
VerifyObjConf.bat	a command file that can be used to verify the configuration of the obj.conf file for the server

Do not move the ServletExec NSAPI directory after installation. There is a registry entry that allows ServletExec to find this directory (see the discussion of registry entries, below); if you move it then ServletExec won't be able to find its configuration files.

Registry Entries

The ServletExec installer creates a new registry entry for each Enterprise server with the following key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\New Atlanta Communications\  
ServletExec NSAPI\https-<server name>
```

Following Netscape convention, the key for FastTrack servers is the same, except that `httpd-<server name>` is used as the final part of the key.

This key contains a single parameter, `Home`, that contains the path to the server's sub-directory within the ServletExec NSAPI directory. If you move the ServletExec NSAPI directory after installation, you must modify the key for each server to contain the new path.

Netscape Configuration File (`obj.conf`)

The Netscape server implementation requires that the **obj.conf** configuration file be modified in order to install NSAPI plug-ins such as ServletExec. This section describes the modifications performed by the ServletExec installer; if you chose not to allow the installer to make these modifications, you must make them manually.

Several lines must be added to the **obj.conf** configuration file for each server for which ServletExec is installed (the location of these lines within the **obj.conf** file is very important):

- 1) The following lines must be added to the beginning of **obj.conf** before the other `Init` directives:

```
Init fn=load-modules shlib="<path>/ServletExec_NSAPI.dll"  
      funcs="ServletExecInit,ServletExecFilter,ServletExecService"  
Init fn=ServletExecInit
```

where `<path>` is the full path to the `ServletExec_NSAPI.dll`. (Note: the first `Init` directive will normally appear on a single line within the **obj.conf** file. It's shown as spanning two lines here for formatting reasons. It may span two lines within **obj.conf**, in which case the second line must begin with a tab or space character).

IMPORTANT! If you had previously activated, then deactivated the Java Interpreter for Enterprise 3.5.1 or 3.6 as described above, two `Init` directives similar to the following will still be in your **obj.conf** file:

```
Init funcs="SJavaBootInit" shlib=".." fn="load-modules"  
Init classpath=".." ldpath=".." fn="SJavaBootInit"
```

You must either: (a) make sure the ServletExec Init directives appear before these two Java Interpreter directives; or, (b) remove the Java Interpreter directives.

- 2) The following lines must be added within the <Object name=default> directives:

```
NameTrans fn=ServletExecFilter root="<document root>"
Service method=(GET|HEAD|POST) type=magnus-internal/nac
      fn=ServletExecService
```

where <document root> is the full path to the server's document root directory. This will be the same as the directory specified in the fn=document-root directive provided by Netscape. (Note: the Service directive will normally appear on a single line within the **obj.conf** file. It's shown as spanning two lines here for formatting reasons. It may span two lines within **obj.conf**, in which case the second line must begin with a tab or space character).

IMPORTANT! The NameTrans directive for ServletExec must appear first in the list of NameTrans directives for the default object. For Enterprise 3.5.1 and 3.6, the Service directive for ServletExec must similarly appear first in the list of Service directives for the default object.

IMPORTANT! If hardware virtual servers are being used, for each hardware virtual server the following NameTrans directive must be placed before the ServletExec NameTrans directive described above:

```
NameTrans fn=ServletExecFilter address="<IP address>"
      root="<server document root>"
```

where <server document root> is the document root directory for the virtual server. This will be the same as the directory specified in the fn=document-root directive provided by Netscape for the virtual server. See Chapter 2 for a discussion of hardware virtual servers.

Figure B-1 on page B-9 shows the complete **obj.conf** file for FastTrack/Enterprise 3.0 with ServletExec installed. Figure B-3 on page B-10 shows a partial **obj.conf** file for Enterprise 3.5.1 with ServletExec installed. The default installation locations were used for both the server and ServletExec in both cases. The changes made for ServletExec are highlighted.

Figure B-2 on page B-9 shows a portion of an **obj.conf** file showing the NameTrans directives for a hardware virtual server. The changes made for ServletExec are highlighted.

VerifyObjConf.bat

Within the server sub-directory of the ServletExec NSAPI directory is the command file VerifyObjConf.bat. This command file can be executed to automatically check your **obj.conf** file for errors. VerifyObjConf.bat creates a file named Verify.log in the server subdirectory that contains a list of warnings and errors.

JDK/JRE Installation

In order to install ServletExec 2.2 you must have first installed either JDK or JRE version 1.1.x or 1.2. *The JDK or JRE must be installed on a local drive and not on a mapped network drive.* If the JDK is installed on a mapped network drive, ServletExec will not be able to load and initialize the Java VM.

It's possible you may have multiple versions of the JDK installed on your system, or that you may install newer (or older) versions after installing ServletExec. ServletExec uses registry entries to determine which version of the JDK to use. It will look for an installed JDK first and if it doesn't find one it will look for the JRE. Here's the complete algorithm:

1. Look for the following registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Development Kit

If found, go to step 2, otherwise look for the following key:

HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Runtime Environment

2. Read the CurrentVersion variable from the key found in step 1. Currently, the only valid values for this variable start with "1.1" or "1.2" (including, for example "1.1.7").
3. Append the value of the CurrentVersion variable from step 2 to the key from step 1 to create a new key. For example:

HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Development Kit\1.1

or

HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Runtime Environment\1.1

4. Read the value of the JavaHome variable for the key from step 3 to find the location of the JDK or JRE.

If you launch DBMON and then restart your web server, ServletExec displays the Java VM settings in the DBMON console window during initialization (see Appendix E for a discussion of DBMON). You can examine the classpath displayed by ServletExec to see which version of the JDK is being used.

Uninstalling ServletExec

Perform the following steps to uninstall ServletExec for a Netscape FastTrack or Enterprise server:

1. Open the Control Panel and select Add/Remove Programs.
2. Select "ServletExec NSAPI 2.2 (<server name>)" and click Add/Remove.
3. The uninstaller may not be able to remove all of the installed files, so remove the <server name> directory from within the ServletExec NSAPI directory. The de-

fault location of the ServletExec NSAPI directory is within the C:\Netscape\SuiteSpot\plugins directory.

4. Edit the **obj.conf** configuration file for the server and remove the lines that were added for ServletExec (as described above).
5. If ServletExec is being completely removed from your system, delete the Servlet-Exec NSAPI directory after completing steps 1 through 4 for all servers for which ServletExec was installed.

```

Init fn=load-modules
    shlib="C:/Netscape/SuiteSpot/plugins/ServletExec_NSAPI/ServletExec_NSAPI.dll"
    funcs="ServletExecInit,ServletExecFilter,ServletExecService"
Init fn=ServletExecInit
Init fn=flex-init access="C:/Netscape/SuiteSpot/https-pooH/logs/access"
format.access="%Ses->client.ip% - %Req->vars.auth-user% [%SYSDATE%] \
    \"%Req->reqpb.clf-request% \
    \" %Req->srvhdrs.clf-status% %Req->srvhdrs.content-length%\"
Init fn=load-types mime-types=mime.types

<Object name=default>
NameTrans fn=ServletExecFilter root="C:/Netscape/SuiteSpot/docs"
NameTrans fn=pfx2dir from=/ns-icons dir="C:/Netscape/SuiteSpot/ns-icons"
NameTrans fn=pfx2dir from=/mc-icons dir="C:/Netscape/SuiteSpot/ns-icons"
NameTrans fn=document-root root="C:/Netscape/SuiteSpot/docs"
PathCheck fn=nt-uri-clean
PathCheck fn="check-acl" acl="default"
PathCheck fn=find-pathinfo
PathCheck fn=find-index index-names="index.html,home.html"
ObjectType fn=type-by-extension
ObjectType fn=force-type type=text/plain
Service method=(GET|HEAD|POST) type=magnus-internal/nac fn=ServletExecService
Service method=(GET|HEAD) type=magnus-internal/imagemap fn=imagemap
Service method=(GET|HEAD) type=magnus-internal/directory fn=index-common
Service method=(GET|HEAD) type=*-magnus-internal/* fn=send-file
AddLog fn=flex-log name="access"
</Object>

<Object name=cgi>
ObjectType fn=force-type type=magnus-internal/cgi
Service fn=send-cgi
</Object>

```

Figure B-1. Complete obj.conf Configuration File for FastTrack/Enterprise 3.0

```

.
.
<Object name=default>
NameTrans fn=ServletExecFilter address="168.121.97.173"
    root="D:/Netscape/SuiteSpot/docs3"
NameTrans fn=ServletExecFilter root="D:/Netscape/SuiteSpot/docs"
NameTrans fn=pfx2dir from=/ns-icons dir="D:/Netscape/SuiteSpot/ns-icons"
NameTrans fn=pfx2dir from=/mc-icons dir="D:/Netscape/SuiteSpot/ns-icons"
NameTrans fn=document-root address="168.121.97.173" root="D:/Netscape/SuiteSpot/docs3"
NameTrans fn=document-root root="D:/Netscape/SuiteSpot/docs"
.
.

```

Figure B-2. NameTrans Directives for Hardware Virtual Server

```

Init fn="load-modules"
    shlib="C:/Netscape/SuiteSpot/plugins/ServletExec_NSAPI/ServletExec_NSAPI.dll"
    funcs="ServletExecInit,ServletExecFilter,ServletExecService"
Init fn="ServletExecInit"
Init fn=flex-init access="C:/Netscape/SuiteSpot/https-ntserver1/logs/access"
format.access="%Ses->client.ip% - %Req->vars.auth-user% [%SYSDATE%] \
    \"%Req->reqpb.clf-request% \
    \" %Req->srvhdrs.clf-status% %Req->srvhdrs.content-length%"
Init fn=load-types mime-types=mime.types
Init fn="load-modules"
    funcs="CM_Init,CM_Delete,CM_Index,CM_Get,CM_Put,CM_Move,CM_Mkdir,CM_Post,CM_Edit,
CM_Unedit,CM_Save,CM_Lock,CM_Unlock,CM_RevLabel,CM_RevLog,CM_SetAttr,CM_GetAttr,
CM_GetAttrNames,CM_RevAdd,CM_RevNum,CM_Copy,CM_GetPS,CM_StartRev,CM_StopRev"
    shlib="C:/Netscape/SuiteSpot/plugins/content_mgr/bin/content_mgr.dll"
    NativeThread="no"
Init fn="CM_Init" webconfig="C:/Netscape/SuiteSpot/https-ntserver1/config/webpub.conf"
Init fn="load-modules" funcs="es-search-init,es-search,es-search-nametrans"
    shlib="C:/Netscape/SuiteSpot/plugins/search/bin/nsir.dll" NativeThread="no"
Init fn="es-search-init"
    systemini="C:/Netscape/SuiteSpot/httpsntserver1/config/webpub.conf"
    errordb="C:/Netscape/SuiteSpot/plugins/search/admin/nsir.err"
    userdefsdb="C:/Netscape/SuiteSpot/plugins/search/admin/userdefs.ini"
Init fn="load-modules"
    funcs="ns_agentInit,agent_name_trans,ns_agentCmdHandler,ns_agentType"
    shlib="C:/Netscape/SuiteSpot/plugins/agents/bin/agents.dll" NativeThread="no"
Init fn="ns_agentInit"
    systemini="C:/Netscape/SuiteSpot/plugins/agents/data/agent_system.ini"
    uidir="C:/Netscape/SuiteSpot/plugins/agents/data" LateInit="yes"

<Object name=default>
NameTrans fn="ServletExecFilter" root="C:/Netscape/SuiteSpot/docs"
NameTrans fn=pfx2dir from=/ns-icons dir="C:/Netscape/SuiteSpot/ns-icons"
NameTrans fn=pfx2dir from=/mc-icons dir="C:/Netscape/SuiteSpot/ns-icons"
NameTrans fn="pfx2dir" from=/help dir="C:/Netscape/SuiteSpot/manual/https/ug"
NameTrans from=/Agents fn="agent_name_trans"
NameTrans fn="pfx2dir" from="/search-ui" dir="C:/Netscape/SuiteSpot/plugins/search/ui"
NameTrans fn="es-search-nametrans" from="/search"
NameTrans fn="pfx2dir" from="/webpub-ui"
    dir="C:/Netscape/SuiteSpot/plugins/content_mgr/ui"
NameTrans fn="pfx2dir" from="/publisher"
    dir="C:/Netscape/SuiteSpot/plugins/content_mgr/client"
NameTrans fn=document-root root="C:/Netscape/SuiteSpot/docs"
PathCheck fn=nt-uri-clean
PathCheck fn="check-acl" acl="default"
PathCheck fn=find-pathinfo
PathCheck fn=find-index index-names="index.html,home.html"
ObjectType fn=type-by-extension
ObjectType fn=force-type type=text/plain
Service method="(GET|HEAD|POST)" type="magnus-internal/nac" fn="ServletExecService"
Service method=(GET|HEAD) type=magnus-internal/imapmap fn=imapmap
Service method=(GET|HEAD) type=magnus-internal/directory fn=index-common
Service fn="CM_Delete" method="DELETE"
Service fn="CM_Index" method="INDEX"
Service fn="CM_Get" method="(GET|HEAD)"
Service fn="CM_Put" method="PUT"
Service fn="CM_Move" method="MOVE"
Service fn="CM_Mkdir" method="MKDIR"
Service fn="CM_Post" method="POST"
Service fn="CM_Copy" method="COPY"
Service fn="CM_Edit" method="EDIT"
.
.

```

Figure B-3. Partial obj.conf Configuration File for Enterprise 3.5.1 and 3.6



Appendix B2. Netscape/NSAPI Installation (SPARC Solaris)

Verifying Your ServletExec Installation

This appendix contains important information that will allow you to verify your installation of ServletExec NSAPI for Netscape FastTrack & Enterprise servers for SPARC Solaris. It will also be useful if you decide to uninstall ServletExec to make sure you've completely removed all installed components.

In order to complete the ServletExec installation, the Netscape **obj.conf** configuration file and server **start** script must be modified. The ServletExec 2.2 installation script gives you the option to allow it to automatically update these files. Instructions in this appendix allow you to verify these changes, or to manually edit **obj.conf** and the **start** script if you chose not to let the installation script do so automatically.

ServletExec NSAPI for Netscape FastTrack & Enterprise servers on SPARC Solaris is implemented using Netscape's NSAPI server programming interface, and the standard Java Virtual Machine (VM) from Sun Microsystems. This combination gives you maximum performance and Java compatibility. ServletExec NSAPI runs as an in-process extension of your web server.

Upgrading from a Previous Version

If you've installed ServletExec for multiple Netscape server instances, you only need to upgrade ServletExec for one instance and the upgrade will apply to all instances. In order to upgrade to a new version of ServletExec and maintain your old configuration settings, perform the following steps:

1. Stop your web server. If you've installed ServletExec for multiple Netscape server instances, stop them all.
2. Make backup copies of the ServletExecData and Servlets directories for the Netscape server instance you're upgrading. The default location for these directories is within the `netscape/suitespot/plugins/ServletExecNSAPI/https-<server name>` directory.

3. Run the installer for the new version of ServletExec while logged in as root, entering the path to the Netscape server installation directory when prompted. The installer will overwrite all of the appropriate ServletExec files previously installed in the ServletExecNSAPI directory. The installer will not modify the contents of the `https-<server name>` sub-directories.
4. Restart your web server.

After restarting your web server, the new version of ServletExec should run using your old configuration settings. If you have any problems, you can restore the `ServletExecData` and `Servlets` sub-directories from the backups you made in step 2, above.

System Requirements

- ServletExec 2.2 requires SPARC Solaris version 2.5.1 (with a kernel patch), 2.6 (with a patch), or 7. The patch for Solaris 2.5.1 can be downloaded from:

<ftp://sunsolve.sun.com/pub/patches/104283.readme>

<ftp://sunsolve.sun.com/pub/patches/104283-04.tar.Z>

The patch for Solaris 2.6 can be downloaded from:

<ftp://sunsolve.sun.com/pub/patches/105181.readme>

<ftp://sunsolve.sun.com/pub/patches/105181-06.tar.Z>

- ServletExec 2.2 supports FastTrack & Enterprise servers version 3.0 and higher, including Enterprise 3.5, 3.5.1, and 3.6.
- Before installing ServletExec 2.2 you must first install either the Java Development Kit (JDK) or the Java Runtime Environment (JRE) version 1.1.x or 1.2. JDK or JRE 1.1.x can be downloaded from Sun Microsystems' web site:

<http://java.sun.com/products/jdk/1.1/index.html>

<http://java.sun.com/products/jdk/1.1/jre/index.html>

JDK or JRE 1.2 can be downloaded from:

<http://java.sun.com/products/jdk/1.2/index.html>

<http://java.sun.com/products/jdk/1.2/jre/index.html>

Note that Sun has created two versions of the JDK for Solaris, which it refers to as the Production and Reference implementations (or releases). At the time of this writing, the latest versions of the Production and References releases are 1.1.7_05 and 1.2_01. *We recommend using only the latest Production releases of the JDK with ServletExec.* We do not recommend using Reference releases.

If you're using the 1.1.6 Reference implementation of the JDK, you must also download and install the optional Solaris Native Threads Pack after installing the JDK. The Production release of JDK 1.1.5 and greater includes native thread support.

Regardless of whether you're using the Reference or Production version of the JDK, if you're running Solaris 2.5.1 you must also apply two patches to solve native thread synchronization problems. These patches are not required for Solaris 2.6 or 7.

A comparison of the Reference and Production releases of the JDK, and instructions for downloading the Solaris 2.5.1 native threads patches is available from Sun Microsystems' web site:

<http://java.sun.com/products/jdk/1.1/solaris-product-comparison.html>

WARNING! The Reference release of JDK 1.1.5 contains a memory leak that makes it unsuitable for use with ServletExec in production environments. *Do not use the Reference release of JDK 1.1.5 with ServletExec!* The Production releases of JDK 1.1.5 and later does not have this memory leak and may be used with ServletExec.

Uninstall Other Servlet Engines

VERY IMPORTANT! You **must** uninstall any other servlet engines you may have previously installed for use with Netscape FastTrack/Enterprise servers before installing and using ServletExec. In particular, modifications you may have made to the **obj.conf** configuration file for other servlet engines must be removed.

Installing ServletExec for Multiple Servers

If you've created multiple server instances, you must run the ServletExec installer separately for each server for which you want to install ServletExec. Pay attention to the following items when installing ServletExec for multiple servers:

1. Stop all servers before running the ServletExec installer.
2. Use the same installation directory for all ServletExec installations.

The ServletExec installer will create a separate sub-directory within the ServletExecNSAPI directory for each server instance (see further discussion, below).

When installed for multiple Netscape server instances, each instance of ServletExec creates its own Java VM. Therefore, the servlets being hosted by ServletExec for each server are isolated from the others in separate VMs.

Note that you must purchase a unique ServletExec serial number for each Netscape server instance with which ServletExec will be installed.

Deactivate the Java Interpreter

For Enterprise 3.5.1 and 3.6, the built-in Java interpreter is deactivated by default; however, if it has been activated on your server you must deactivate it manually by performing the following steps:

1. From the Netscape Administration Server home page, select the server to be administered.
2. Select “Programs” from the menu bar in the upper frame of the server administration page.
3. Select the “Java” link from the menu in the left frame of the “Programs” page.
4. Turn off the Java interpreter.

This configuration step only applies to Enterprise 3.5.1 and 3.6. If you still have problems running ServletExec after deactivating the Java interpreter, check the `Init` directives in your **obj.conf** file as described below.

What Was Installed?

When you installed ServletExec, these changes were made to your system:

- the `ServletExecNSAPI` directory was created
- the Netscape server **start** script was updated (if you chose to let the ServletExec installation script modify this file)
- the Netscape **obj.conf** configuration file was updated (if you chose to let the ServletExec installation script modify this file)

The following sections describe each of these changes.

The ServletExecNSAPI Directory

The `ServletExecNSAPI` directory was created within the `netscape/suitespot/plugins` directory. The `ServletExecNSAPI` directory contains the following files:

- `ServletExecNSAPI.so`
- JSDK License Agreement
- ServletExec License Agreement
- READ ME
- Release Notes

In addition to the files listed above, the ServletExecNSAPI directory contains the following sub-directories:

Documentation	contains this manual, a tutorial for writing servlets, and the Java Servlet API documentation from the Java Servlet Development Kit (JSDK) 2.1
lib	contains the <code>servlet.jar</code> and <code>ServletExec22.jar</code> files; these archives are automatically added to the ServletExec classpath, and are required for ServletExec operation; DO NOT MOVE OR DELETE THIS DIRECTORY OR THESE FILES
Examples	contains examples of some of ServletExec's advanced features, including: Server-Side Includes (SSI), Page Compilation, JavaServer Pages (JSP), File Upload servlet, and Presentation Templates

Also within the ServletExecNSAPI directory are sub-directories for each server. Following Netscape conventions, the sub-directories are named `https-<server name>` (for Enterprise) or `httpd-<server-name>` (for FastTrack). Within each server sub-directory are the following directories:

classes	this directory is automatically added to the ServletExec classpath; servlet or non-servlet classes can be placed in this directory; see the READ ME for more information
ServletLogs	contains the <code>Servlet.log</code> files; entries are added to the <code>Servlet.log</code> file whenever a servlet invokes its <code>log()</code> method; see the section in Chapter 2 on configuring the servlet logging feature
ServletExecData	contains ServletExec configuration files; DO NOT MODIFY THE CONTENTS OF THIS DIRECTORY ; in emergency situations you may delete this entire directory and ServletExec will recreate it using its default configuration; however, if you do this all changes you've made to the default configuration will be lost and you will be required to re-enter your ServletExec serial number
Servlets	the default location for servlet class files; initially contains sample servlets that ship with ServletExec

Do not move the ServletExecNSAPI directory after installation.

Server "start" Script

The Netscape server **start** script must be modified to set the `LD_LIBRARY_PATH`, `JNI_VERSION`, and `LD_PRELOAD` environment variables, and possibly export the `SERVER_ROOT` environment variable. The start script is located in the `https-<server name>` directory (for Enterprise) or `httpd-<server-name>` directory (for FastTrack).

This section describes the modifications made to the server **start** script by the ServletExec installation script. If you chose not to allow the ServletExec installation script to make these changes, you must make them manually. *If you choose to run a different version of the JDK or JRE than the one available when ServletExec was originally installed you may need to modify the **start** script and change the setting of LD_LIBRARY_PATH.*

LD_LIBRARY_PATH

The LD_LIBRARY_PATH environment variable must be set to include the SPARC native threads libraries (ServletExec *requires* SPARC native threads). For example, if the JDK is installed in /usr/java, then LD_LIBRARY_PATH must be set as follows (for Bourne shells) in the server **start** script depending on which version of the JDK or JRE you're using.

For JDK 1.1.x and JRE 1.1.x production releases:

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/java/lib/sparc/native_threads
export LD_LIBRARY_PATH
```

For JDK 1.2 production releases:

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/java/jre/lib/sparc
export LD_LIBRARY_PATH
```

For JRE 1.2 production releases:

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/java/lib/sparc
export LD_LIBRARY_PATH
```

For JDK 1.2 reference releases:

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/java/jre/lib/sparc
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/java/jre/lib/sparc/native_threads
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/java/jre/lib/sparc/classic
export LD_LIBRARY_PATH
```

For JRE 1.2 reference releases:

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/java/lib/sparc
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/java/lib/sparc/native_threads
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/java/lib/sparc/classic
export LD_LIBRARY_PATH
```

JNI_VERSION

The JNI_VERSION must be set to either “1.1” or “1.2” depending on which version of the JDK is installed. If this environment variable isn't set, then ServletExec assumes JNI version 1.1. Here's an example for JDK 1.2:

```
JNI_VERSION=1.2
export JNI_VERSION
```

LD_PRELOAD

The LD_PRELOAD environment variable must be set to explicitly load libjava.so before any other shared objects:

```
LD_PRELOAD=libjava.so
export LD_PRELOAD
```

(Some ServletExec users have reported the inability to run CGI programs after installing ServletExec. In these cases, removing these LD_PRELOAD statements from the **start** script has been known to resolve the problem. However, the side effect of removing these statements is that Java VM's JITC may not work properly).

SERVER_ROOT

Because the Netscape server installation directory may vary, the **SERVER_ROOT** environment variable may need to be exported in the **start** script to direct ServletExec to that location. This is only required if your Netscape server installation is anywhere other than `/usr/netscape/suitespot` (which ServletExec will assume as the default). If your installation uses a base installation directory other than the default, the **start** script *must* contain the following line:

```
export SERVER_ROOT
```

Netscape Configuration File (obj.conf)

The Netscape server implementation requires that the **obj.conf** configuration file be modified in order to install NSAPI plug-ins such as ServletExec. This section describes the modifications performed by the ServletExec installation script; if you chose not to allow the installation script to make these modifications, you must make them manually.

Several lines must be added to the **obj.conf** configuration file for each server for which you've installed ServletExec (the location of these lines within the **obj.conf** file is very important):

- 1) The following lines must be added to the beginning of **obj.conf** before the other **Init** directives:

```
Init fn=load-modules shlib="<path>/ServletExecNSAPI.so"
    funcs="ServletExecInit,ServletExecFilter,ServletExecService"
Init fn=ServletExecInit
```

where `<path>` is the full path to the `ServletExecNSAPI.so` file. (Note: the first **Init** directive will normally appear on a single line within the **obj.conf** file. It's shown as spanning two lines here for formatting reasons. It may span two lines within **obj.conf**, in which case the second line must begin with a tab or space character).

IMPORTANT! If you had previously activated, then deactivated the Java Interpreter for Enterprise 3.5.1 or 3.6 as described above, two **Init** directives similar to the following will still be in your **obj.conf** file:

```
Init funcs="SJavaBootInit" shlib="..." fn="load-modules"
Init classpath="..." ldpath="..." fn="SJavaBootInit"
```

You must either: (a) make sure the ServletExec **Init** directives appear before these two Java Interpreter directives; or, (b) remove the Java Interpreter directives.

- 2) The following lines must be added within the `<Object name=default>` directives:

```
NameTrans fn=ServletExecFilter root="<document root>"
```

```
Service method=(GET|HEAD|POST) type=magnus-internal/nac
      fn=ServletExecService
```

where `<document root>` is the full path to the server's document root directory. This will be the same as the directory specified in the `fn=document-root` directive provided by Netscape. (Note: the `Service` directive will normally appear on a single line within the **obj.conf** file. It's shown as spanning two lines here for formatting reasons. It may span two lines within **obj.conf**, in which case the second line must begin with a tab or space character).

IMPORTANT! The `NameTrans` directive for `ServletExec` must appear first in the list of `NameTrans` directives for the default object. For Enterprise 3.5.1 and 3.6, the `Service` directive for `ServletExec` must similarly appear first in the list of `Service` directives for the default object.

IMPORTANT! If hardware virtual servers are being used, then for each hardware virtual server the following `NameTrans` directive must be placed before the `ServletExec` `NameTrans` directive described above:

```
NameTrans fn=ServletExecFilter address="<IP address>"
      root="<server document root>"
```

where `<server document root>` is the document root directory for the virtual server. This will be the same as the directory specified in the `fn=document-root` directive provided by Netscape for the virtual server. See Chapter 2 for a discussion of hardware virtual servers.

Figure B-1 on page B-9 shows the complete **obj.conf** file for FastTrack/Enterprise 3.0 with `ServletExec` installed. Figure B-3 on page B-10 shows a partial **obj.conf** file for Enterprise 3.5.1 or 3.6 with `ServletExec` installed. The default installation locations were used for both the server and `ServletExec`. The changes made for `ServletExec` are highlighted.

Figure B-2 on page B-9 shows a portion of an **obj.conf** file showing the `NameTrans` directives for a hardware virtual server. The changes made for `ServletExec` are highlighted.

Uninstalling ServletExec

1. Undo the manual configuration steps described above. In particular, remove the entries added to **obj.conf**.
2. Delete the `ServletExecNSAPI` directory.



Appendix B3. Netscape/WAI Installation & Operation

Verifying Your ServletExec Installation Starting and Stopping ServletExecWAI

This appendix contains important information that will allow you to verify your installation of ServletExec WAI for Netscape Enterprise servers, and contains instructions for starting and stopping the ServletExecWAI application. It will also be useful if you decide to uninstall ServletExec to make sure you've completely removed all installed components.

In order to complete the ServletExec installation, the Netscape **obj.conf** configuration file must be modified. The ServletExec 2.2 installation script gives you the option to allow it to automatically update the **obj.conf** file. Instructions in this appendix allow you to verify these changes, or to manually edit **obj.conf** if you chose not to let the installation script do so automatically.

ServletExec WAI for Netscape Enterprise servers is implemented using Netscape's WAI server programming interface, and any JDK 1.1- or 1.2-compliant Java VM. This combination gives you maximum safety, deployment flexibility, and Java compatibility.

In contrast to ServletExec NSAPI versions, which run as in-process extensions of the web server, ServletExec WAI runs as an out-of-process application that communicates with Enterprise server via CORBA/IIOP. This implementation allows ServletExec to run on a remote machine from the Netscape Enterprise server. Instructions for running ServletExec on a remote machine are provided under the heading Remote Operation, below.

Upgrading from a Previous Version

If you've installed ServletExec for multiple Netscape server instances, you only need to upgrade ServletExec for one instance and the upgrade will apply to all instances. In order to upgrade to a new version of ServletExec and maintain your old configuration settings, perform the following steps:

1. Stop the ServletExecWAI application and your web server. If you've installed ServletExec for multiple Netscape server instances, stop them all.

2. Make backup copies of the ServletExecData and Servlets directories for the Netscape server instance you're upgrading. The default location for these directories is within the netscape/suitespot/plugins/ServletExecWAI/https-<server name> directory.
3. Run the installer for the new version of ServletExec while logged in as root, entering the path to the Netscape server installation directory when prompted. The installer will overwrite all of the appropriate ServletExec files previously installed in the ServletExecWAI directory. The installer will not modify the contents of the https-<server name> sub-directories.
4. Restart your web server.

After restarting your web server, the new version of ServletExec should run using your old configuration settings. If you have any problems, you can restore the ServletExecData and Servlets sub-directories from the backups you made in step 2, above.

System Requirements

- ServletExec 2.2 for Netscape/WAI should run on any operating system that supports a JDK 1.1- or 1.2-compliant Java VM and is supported by Netscape Enterprise 3.0, 3.5.1, or 3.6. However, ServletExec 2.2 for Netscape/WAI has only been tested on the following operating systems:
 - HP-UX 10.20 and 11.0
 - SPARC Solaris 2.5.1, 2.6, and 7

These are the only operating systems for which we can “officially” provide technical support. However, we have customers who are successfully running ServletExec WAI on the IRIX, AIX, and Digital Unix operating systems. If you try using ServletExec on one of these operating systems and run into problems, contact us for technical support and we'll do our best to help.

- ServletExec 2.2 supports Netscape Enterprise server version 3.0 and higher, including Enterprise 3.5.1 and 3.6. For Enterprise 3.0, the version 3.01 patch for WAI must be installed. Information on this patch and instructions for installing it can be retrieved from Netscape's web site:

<http://help.netscape.com/filelib.html#wai>

- Before installing ServletExec you must first install either the Java Development Kit (JDK) or the Java Runtime Environment (JRE) version 1.1.x or 1.2. The JDK or JRE for SPARC Solaris can be downloaded from Sun Microsystems' web site:

<http://java.sun.com/products/jdk/1.1/index.html>

<http://java.sun.com/products/jdk/1.1/jre/index.html>

JDK or JRE 1.2 for SPARC Solaris can be downloaded from:

<http://java.sun.com/products/jdk/1.2/index.html>

<http://java.sun.com/products/jdk/1.2/jre/index.html>

The JDK for HP-UX can be downloaded from Hewlett-Packard's web site:

http://www.internetsolutions.enterprise.hp.com/2_60_index.html

Uninstall Other Servlet Engines

VERY IMPORTANT! You **must** uninstall any other servlet engines you may have previously installed for use with Netscape Enterprise server before installing and using ServletExec. In particular, modifications you may have made to the **obj.conf** configuration file for other servlet engines must be removed.

Installing ServletExec for Multiple Servers

If you've created multiple server instances, you must run the ServletExec installer separately for each server for which you want to install ServletExec. Pay attention to the following items when installing ServletExec for multiple servers:

1. Stop all servers before running the ServletExec installer.
2. Use the same installation directory for all ServletExec installations.

The ServletExec installer will create a separate sub-directory within the ServletExecWAI directory for each server instance (see further discussion, below).

When installed for multiple Netscape server instances, each instance of ServletExec uses its own Java VM. Therefore, the servlets being hosted by ServletExec for each server are isolated from the others in separate VMs.

Note that you must purchase a unique ServletExec serial number for each Netscape server instance with which ServletExec will be installed.

Deactivate the Java Interpreter

For Enterprise 3.5.1 and 3.6, the built-in Java interpreter is deactivated by default; however, if it has been activated on your server you must deactivate it manually by performing the following steps:

1. From the Netscape Administration Server home page, select the server to be administered.
2. Select "Programs" from the menu bar in the upper frame of the server administration page.

3. Select the “Java” link from the menu in the left frame of the “Programs” page.
4. Turn off the Java interpreter.

This configuration step only applies to Enterprise 3.5.1 and 3.6. If you still have problems running ServletExec after deactivating the Java interpreter, check the `Init` directives in your **obj.conf** file as described below.

Enable Netscape WAI

For Enterprise 3.0, the version 3.01 patch for WAI must be installed. Information on this patch and instructions for installing it can be retrieved from Netscape’s web site:

<http://help.netscape.com/filelib.html#wai>

In order to run ServletExec WAI (or other WAI applications), enable WAI for your server:

1. From the Netscape Administration Server home page, select the server to be administered.
2. Select “Programs” from the menu bar in the upper frame of the server administration page.
3. Select the “WAI Management” link from the menu in the left frame of the “Programs” page.
4. Under “Enable WAI Services”, select Yes, then click OK.
5. Click “Save and Apply” to save your changes.

What Was Installed?

When you installed ServletExec, these changes were made to your system:

- the ServletExecWAI directory was created
- the Netscape **obj.conf** configuration file was updated (if you chose to let the ServletExec installation script modify this file)

The following sections describe each of these changes.

The ServletExecWAI Directory

The ServletExecWAI directory was created within the Netscape/SuiteSpot/Plugins directory.

The ServletExecWAI directory contains the License Agreement, READ ME, and Release Notes. A **run<server name>** shell script that can be used to start the ServletExecWAI application is placed in the ServletExecWAI directory for each server for

which ServletExec is installed. See the heading ServletExecWAI Operation, below, for further discussion of this script.

Also within the ServletExecWAI directory are the following sub-directories:

Documentation	contains this manual, a tutorial for writing servlets, and the Java Servlet API documentation from the Java Servlet Development Kit (JSDK) 2.1
lib	contains the ServletExecWAI.jar, servlet.jar, and ServletExec22.jar files; the first archive contains the ServletExecWAI application, the latter two archives contain classes that are required by the ServletExecWAI application and are added to the ServletExec classpath in the run<server name> shell script (see discussion below)
Examples	contains examples of some of ServletExec's advanced features, including: Server-Side Includes (SSI), Java Invoker (dynamic page compilation), JavaServer Pages (JSP), File Upload servlet, and Presentation Templates

Also within the ServletExecWAI directory are sub-directories for each server. Following Netscape conventions, the sub-directories are named **https-<server name>**. Within each server sub-directory are the following directories:

ServletLogs	contains the Servlet.log files; entries are added to the Servlet.log file whenever a servlet invokes its log() method; see the section in Chapter 2 on configuring the servlet logging feature
ServletExecData	contains ServletExec configuration files; DO NOT MODIFY THE CONTENTS OF THIS DIRECTORY ; in emergency situations you may delete this entire directory and ServletExec will recreate it using its default configuration; however, if you do this all changes you've made to the default configuration will be lost and you will be required to re-enter your ServletExec serial number
Servlets	the default location for servlet class files; initially contains sample servlets that ship with ServletExec

Do not move the ServletExecWAI directory after installation.

Netscape Configuration File (obj.conf)

The Netscape server implementation requires that the **obj.conf** configuration file be modified in order to install WAI applications such as ServletExec. This section describes the modifications performed by the ServletExec installation script; if you chose not to allow the installation script to make these modifications, you must make them manually.

Several lines must be added to the **obj.conf** configuration file for each server for which you've installed ServletExec (the location of these lines within the **obj.conf** file is very important):

- 1) The following lines must be added to the beginning of **obj.conf** before the other NameTrans directives:

```
NameTrans fn="assign-name" from="/servlet/*" name="servletexec"  
NameTrans fn="assign-name" from="*.shtml" name="servletexec"  
NameTrans fn="assign-name" from="*.jhtml" name="servletexec"
```

Servlet aliases: for each servlet alias that is configured via the ServletExec Admin UI, a NameTrans directive must be added to **obj.conf** with the following format:

```
NameTrans fn="assign-name" from="<servlet alias>" name="servletexec"
```

For suffix aliases, the <servlet alias> in the NameTrans directive is exactly the same as the Alias configured in the ServletExec Admin UI. For prefix aliases, the <servlet alias> is the same as configured in the ServletExec Admin UI with an additional trailing asterisk. For example:

```
NameTrans fn="assign-name" from="/date*" name=servletexec
```

See Chapter 2 for a discussion of Servlet Aliases. Note that whenever changes are made to **obj.conf** the server must be restarted for the changes to take effect.

- 2) At the end of **obj.conf** an object name "servletexec" must be added using the following directives:

```
<Object name="servletexec">  
Service fn="IIOPexec" name="ServletExecWAI"  
</Object>
```

ServletExecWAI Operation

ServletExecWAI is a standalone Java application that communicates with the Enterprise server via CORBA. ServletExecWAI is started using the java command on the command-line; in addition to java command arguments, ServletExecWAI accepts several required and optional arguments.

The ServletExec installation script places a **run<server name>** shell script in the ServletExecWAI directory to help automate execution of the java command to start the ServletExecWAI application. The next sections describe use of this script and manual execution of the java command.

run<server name> Shell Script

The **run<server name>** shell script performs the operations described in the next section for manual startup and provides reasonable defaults for all of the ServletExecWAI run-

time arguments. The shell script provides command line options to override any of the defaults:

```
run<server name> [options]
```

options:

```
-h : show this Help
-i <hostname:port> or <IP address:port> or <path to ".IOR" file>
-n <ServletExecWAI application name>
-d <ServletExecWAI home directory>
-m <full path to mime types file>
-r <server's document root directory>
-j <Java VM options>
```

The **run<server name>** script is created during ServletExecWAI installation and a separate script is created for each of your Enterprise server instances. The **run<server name>** options map to the ServletExecWAI application arguments as follows:

<u>Run<server> script option</u>	<u>ServletExecWAI argument</u>
-i	-host
-n	-name
-d	-home
-m	-mimetypes
-r	-root

The **-j** option allows you to add options to the java command. If the **-classpath** java command option is specified, the **CLASSPATH** environment variable created in the **run<server name>** script is prepended to the path provided using the **-j** option.

The **run<server name>** scripts do not execute ServletExecWAI in the background and can be used in **inittab** to automatically start/restart ServletExecWAI.

Manual Startup (java Command)

Before starting ServletExecWAI via the command line, you must add the files **ServletExecWAI.jar**, **nisb.zip**, and **WAI.zip** to the **CLASSPATH**; the latter two files are part of the Netscape Enterprise server installation and can be found in the directory **Netscape/SuiteSpot/wai/java**.

Start the ServletExecWAI using the java command as follows:

```
java [java arguments] ServletExecWAI [arguments]
```

The java command arguments aren't discussed here. The ServletExecWAI arguments are:

```
-host <hostname:port> or <IP address:port> or <path to ".IOR" file>
-name <ServletExecWAI application name>
```

```
-home <ServletExecWAI home directory>
-mimetypes <path to Enterprise server mime.types file>
-root <path to server's document root directory>
-root <path to the virtual server's document root directory>
```

If an option value contains spaces then it must be placed inside double quotes.

The `-host` argument is optional. If not specified, ServletExecWAI assumes the Enterprise server is running on the same machine as the ServletExecWAI application. (See the heading Remote Operation, below, if you're running ServletExecWAI on a different machine than the Enterprise server). If specified, the port number is optional and defaults to 80. If ServletExecWAI is running with an SSL server, the `-host` argument takes the parameter "file:" followed by the path to the ".IOR" file for the server. This file is located in the "wai/NameService. For example:

```
file:/usr/netscape/suitespot/wai/NameService/https-<server name>.IOR
```

The `-name` argument is optional; the default value is "ServletExecWAI". This value must match the value of the name parameter in the Service directive of the servletexec object in **obj.conf** (see above).

The `-home` argument is optional; the default is the directory from which the java command is executed.

The `-mimetypes` argument is optional. It specifies the path to the Enterprise server **mime.types** file (this file can be found in the config directory for the server instance). If this argument isn't provided, the `ServletContext.getMimeType()` method always returns null.

The `-root` argument is required. It specifies the path to the server's document root directory. If ServletExecWAI is being used with virtual servers, the document root directory of each virtual server must be configured using the `-root` argument. For example:

```
java ServletExecWAI -root "/opt/Netscape/SuiteSpot/docs" \
    -root www.vs1.com=/opt/Netscape/SuiteSpot/vs1dos
```

If the document root directory of a virtual server isn't specified, the document root for the main server is used.

Stopping ServletExecWAI

The ServletExecWAI application must be stopped either from the Shutdown page of the ServletExec Admin UI (see Chapter 2) or by using the StopSEWAI Java application. If ServletExecWAI is not shutdown properly, `destroy()` servlet methods are not invoked, and buffered log messages are lost.

The StopSEWAI application can be executed using the java command:

```
java StopSEWAI [-options]
```

Options to the StopSEWAI application are:

- help
- host <hostname:port> or <IP address:port>
- admin <username/password>

The `-host` parameter specifies the server on which the ServletExecWAI application is running, and is optional. If not specified, `-host` defaults to the local IP address. If `-host` is specified, the port number is optional and defaults to 80.

The `-admin` parameter specifies the ServletExec Admin user name and password and is required if the ServletExec Admin user name and password has been configured (see Chapter 2 for more information).

Remote Operation

The following steps must be taken to run ServletExecWAI on a remote machine (on a different machine than the Enterprise server):

1. Modify Netscape/WAI to allow remote connections by editing **obj.conf** to change the following line from:

```
Init LateInit="yes" fn="IIOPinit"
```

to:

```
Init LateInit="yes" fn="IIOPinit" OAipaddr="<local IP address>"
```

The `OAipaddr` parameter specifies the address on which the Enterprise server listens for IIOP connections. By default the Enterprise server listens to address 127.0.0.1 which limits it to receiving IIOP connections only from the local machine.

2. Stop and restart the Enterprise server.
3. Place a copy of **mime.types** on the remote machine (the machine on which ServletExecWAI is running). Specify the path to this file using the `-mimetypes` argument to ServletExecWAI.
4. When starting ServletExecWAI, use the `-root` parameter to specify a directory on the remote machine (the machine on which ServletExecWAI is running).

Uninstalling ServletExec

1. Undo the manual configuration steps described above. In particular, remove the entries added to **obj.conf**.
2. Delete the ServletExecWAI directory.

Known Limitations

The `HttpServletRequest.getHeaderNames()` method always returns `null`.

References

Complete documentation on the Netscape WAI interface can be found on Netscape's web site:

<http://developer.netscape.com/docs/manuals/enterprise/wai/contents.htm>



Appendix C1. Apache Installation & Operation (Windows)

Completing and Verifying Your ServletExec Installation Starting and Stopping ServletExecApache

This appendix contains important information that will allow you to complete and verify your installation of ServletExec for Apache on Windows, and contains instructions for starting and stopping the ServletExecApache application. It will also be useful if you decide to uninstall ServletExec to make sure you've completely removed all installed components.

In order to complete the ServletExec installation, you must manually modify the Apache httpd.conf configuration file. Instructions for modifying this file are provided under the heading Manual Configuration, below.

ServletExec for Apache is implemented as an out-of-process application that communicates with the Apache Server via network sockets; a small native-code module is added to the Apache Server to manage this communication. This implementation allows ServletExec to run on a remote machine from the Apache Server. Instructions for running ServletExec on a remote machine are provided under the heading Remote Operation, below.

System Requirements

- ServletExec 2.2 only works with the Apache Server version 1.3.9 on Windows (on UNIX, ServletExec 2.2 works with Apache 1.3.1 and higher).
- In order to install ServletExec 2.2 you must first install either the Java Development Kit (JDK) or the Java Runtime Environment (JRE) version 1.1.x or 1.2 from Sun Microsystems, the IBM JDK or JRE for Windows version 1.1.x, or the Microsoft VM. If you plan to use the Microsoft VM we recommend that you first upgrade to the latest version.

The Sun JDK or JRE 1.1.x for Windows can be downloaded from:

<http://java.sun.com/products/jdk/1.1/index.html>

<http://java.sun.com/products/jdk/1.1/jre/index.html>

The Sun JDK or JRE 1.2 for Windows can be downloaded from:

<http://java.sun.com/products/jdk/1.2/index.html>

<http://java.sun.com/products/jdk/1.2/jre/index.html>

The IBM JDK or JRE 1.1.x for Windows can be downloaded from:

<http://www.ibm.com/developer/java/>

The latest version of the Microsoft VM can be downloaded from:

<http://www.microsoft.com/java/>

At the time of this writing, the latest production versions of the Sun JDKs for Windows are 1.1.8 and 1.2.2; the latest production version of the IBM JDK is 1.1.8; the latest production version of the Microsoft VM is build 3186. We recommend using these versions with ServletExec 2.2.

WARNING! Sun JDK 1.1.5 contains a memory leak, which makes it unsuitable for use with ServletExec in production environments. *Do not use JDK 1.1.5 with ServletExec!*

The JDK or JRE must be installed on a local drive and not on a mapped network drive. If the JDK is installed on a mapped network drive, ServletExec will not be able to load and initialize the Java VM.

Uninstall Other Servlet Engines

VERY IMPORTANT! You **must** uninstall any other servlet engines you may have previously installed for use with the Apache Server before installing and using ServletExec. In particular, modifications you may have made to the **httpd.conf** configuration file for other servlet engines must be removed.

Manual Configuration

In order to complete the ServletExec installation, you must manually modify the Apache **httpd.conf** configuration file. *You must stop and restart the Apache Server after modifying this configuration file.*

Add the following directive to the **httpd.conf** file with the other LoadModule directives (near the top of the file):

```
LoadModule servletexec_module modules/ApacheModuleServletExec.dll
```

This directive causes the Apache Server to load the ServletExec native code module that manages communication with the ServletExecApache Java application.

Add the following lines to the end of the **httpd.conf** file:

```
#
# ServletExec directives
#
<Location /servlet>
    SetHandler servlet-exec
</Location>

AddHandler servlet-exec jhtml
AddHandler servlet-exec jsp
AddHandler servlet-exec.shtml
```

These directives define the default prefix and suffix aliases used by ServletExec (see Chapter 2 for a general discussion of servlet aliases, and further discussion of configuring prefix and suffix aliases with Apache, below).

NOTE: if you plan to use the Apache Server rather than ServletExec to parse server-side includes (SSI) pages using the file extension `.shtml` then don't add the last `AddHandler` directive to the **httpd.conf** file.

Prefix Aliases

For each prefix alias that is configured via the ServletExec Admin UI, a `Location` directive must be added to **httpd.conf** in the following format:

```
<Location /prefix-alias>
    SetHandler servlet-exec
</Location>
```

In the manual configuration instructions, above, ServletExec is configured to handle the `/servlet` prefix alias by default.

Suffix Aliases

There are two methods for configuring suffix aliases with Apache. The most common method is illustrated above. For each suffix alias that is configured via the ServletExec Admin UI, add an `AddHandler` directive to **httpd.conf** with the following format:

```
AddHandler servlet-exec <suffix-alias>
```

This method should be used if the URL Rewriting option of the Session Tracking feature is disabled, which it is by default (see Chapter 4 for a discussion of Session Tracking and URL Rewriting). This method of configuring suffix aliases provides better performance than the alternative method.

If the URL Rewriting option of the Session Tracking feature is enabled, an alternative method of configuring suffix aliases must be used. For this method, a `Location` directive is used instead of the `AddHandler` directive. For each suffix alias that is configured via the ServletExec Admin UI, add a `Location` directive to **httpd.conf** with the following format:

```
<Location /*.suffix-alias*>
    SetHandler servlet-exec
</Location>
```

Also, for each sub-directory level beneath the web server document root directory, an additional Location directive must be added to **httpd.conf**. For example, assume the web server root document directory is C:\Program Files\Apache Group\Apache\htdocs and the following sub-directories appear beneath htdocs:

```
htdocs\subdir1
htdocs\subdir2
htdocs\subdir2\subdir3
```

In this example, there are two levels of sub-directories beneath htdocs, so to create a suffix alias of .jsp, three Location directives needs to be added to **httpd.conf**, one for htdocs and one for each sub-directory level:

```
#
# suffix alias for htdocs
#
<Location /*.jsp*>
    SetHandler servlet-exec
</Location>

#
# suffix alias for htdocs\subdir1 and htdocs\subdir2
#
<Location /*/*.jsp*>
    SetHandler servlet-exec
</Location>

#
# suffix alias for htdocs\subdir2\subdir3
#
<Location /*/**/*.jsp*>
    SetHandler servlet-exec
</Location>
```

What Was Installed?

When you installed ServletExec, these changes were made to your system:

- the ServletExec directory was created
- the file ApacheModuleServletExec.dll was added to the Apache modules directory

The ServletExec Directory

The ServletExec directory was created within the Apache installation directory (the default location of the Apache installation directory is C:\Program Files\Apache Group\Apache). The ServletExec directory contains the following files:

- Configure.txt (contains configuration notes)
- JSDK License Agreement
- ServletExec License Agreement

- **READ ME**
- **Release Notes**
- **ServletExecApache.bat**, which is used for starting the ServletExecApache application; see the heading **ServletExecApache Operation**, below, for further discussion of this batch file

In addition to the files listed above, the ServletExec directory contains the following sub-directories:

classes	this directory is automatically added to the ServletExec classpath; servlet or non-servlet classes can be placed in this directory; see the READ ME for more information
Documentation	contains this manual, a tutorial for writing servlets, and the Java Servlet API documentation from the Java Servlet Development Kit (JSDK) 2.1
lib	contains the ServletExecApache.jar, servlet.jar, and ServletExec22.jar files; the first archive contains the ServletExecApache application, the latter two archives contain classes that are required by the ServletExecApache application and are added to the ServletExec classpath in the ServletExecApache.bat command file (see further discussion, below)
Examples	contains examples of some of ServletExec's advanced features, including: Server-Side Includes (SSI), Page Compilation, JavaServer Pages (JSP), File Upload servlet, and Presentation Templates
Servlet Logs	contains the Servlet.log files; entries are added to the Servlet.log file whenever a servlet invokes its log() method; see the section in Chapter 2 on configuring the servlet logging feature
ServletExec Data	contains ServletExec configuration files; DO NOT MODIFY THE CONTENTS OF THIS DIRECTORY ; in emergency situations you may delete this entire directory and ServletExec will recreate it using its default configuration; however, if you do this all changes you've made to the default configuration will be lost and you will be required to re-enter your ServletExec serial number
Servlets	the default location for servlet class files; initially contains sample servlets that ship with ServletExec

Do not move the ServletExec directory after installation.

ServletExecApache Operation

ServletExecApache is a standalone Java application that communicates with the Apache Server via network sockets. ServletExecApache is started using the `java` command on the command-line; in addition to `java` command arguments, ServletExecApache accepts several required and optional arguments.

The ServletExecApache application can be started using the `ServletExecApache.bat` batch file, or directly from the command line using the `java` command.

ServletExecApache.bat

The batch file `ServletExecApache.bat` is provided for your convenience in starting the ServletExecApache application. `ServletExecApache.bat` includes defaults for all of the `java` and ServletExecApache command-line arguments; edit `ServletExecApache.bat` to modify these command-line arguments. A complete list of ServletExecApache command-line arguments is provided under the heading Manual Startup, below.

Closing the DOS Window

When starting the ServletExecApache application using the `java` or `oldjava` commands (either via the `ServletExecApache.bat` file or from the command line), the DOS window must remain open while ServletExecApache is running. In order to be able to close the DOS window, use the `javaw` or `oldjavaw` commands; after ServletExecApache has started you can close the DOS window and ServletExecApache will continue running.

Manual Startup (java Command)

(Note: this section applies to using the `java`, `javaw`, `oldjava`, and `oldjavaw` commands. Only the `java` command is used in the following discussion for illustrative purposes)

You can start the ServletExecApache application using the `java` command as follows:

```
java [java arguments] ServletExecApache [arguments]
```

The `java` command arguments must include the `-classpath` option specifying the path to the `ServletExecApache.jar` file.

The ServletExecApache arguments are:

- help
- port <port number>
- backlog <length>
- home <ServletExecApache home directory>
- root <path to server's document root directory>
- root <path to the virtual server's document root directory>
- mimetypes <path to Apache Server mime.types file>
- allow <ip1,ip2,...,ipn>

If an option value contains spaces then it must be placed inside double quotes.

The `-port` argument is optional; the default value is 8888. This argument specifies the TCP/IP port on which the ServletExecApache application communicates with the Apache Server. If you modify this value you must also add the following variable to the Apache Server **httpd.conf** file:

```
ServletExecInstances <ip address>:<port>
```

where `<ip address>` is the IP address of the machine on which ServletExecApache is running (127.0.0.1 if ServletExecApache is running on the local machine; see the discussion below for running ServletExecApache on a remote machine), and `<port>` is the port number on which ServletExecApache will communicate. Restart the Apache Server after modifying the **httpd.conf** file.

The `-backlog` argument is optional; the default value is 50. This argument specifies the size of the ServletExecApache incoming request queue.

The `-home` argument is optional; the default is the directory from which the java command is executed. This specifies the directory in which ServletExecApache will look for the ServletExec Data, Servlet Logs, and Servlets sub-directories.

The `-root` argument is required. It specifies the path to the Apache Server's document root directory. If ServletExecApache is being used with virtual servers, the document root directory of each virtual server must be configured using a separate `-root` argument using the following format:

```
-root <virtual server>=<document root directory>
```

for example:

```
-root www.abc.com=C:\Apache\htdocs\abcdocs
```

The `-mimetypes` argument is optional. It specifies the path to the Apache Server **mime.types** file. If this argument isn't provided, the `ServletContext.getMimeType()` method always returns null.

The `-allow` parameter is optional. It specifies the address(es) of the Apache Server(s) that are allowed to communicate with the ServletExecApache application. An IP address can include the "*" character to indicate a subrange (for example: 168.121.97.*).

Stopping ServletExecApache

The ServletExecApache application must be stopped from the Shutdown page of the ServletExec Admin UI (see Chapter 2). If ServletExecApache is not shutdown properly, `destroy()` servlet methods are not invoked, SSI counters are not saved, sessions are not saved, and buffered log messages are lost.

If the Apache Server is not running or you are unable to access the ServletExec Admin UI, you can stop the ServletExecApache application by entering CTRL-C in the DOS window in which ServletExecApache is running. If you started ServletExecApache using

the `javaw` or `oldjavaw` commands then the DOS window isn't available; in this case use the Windows NT Task Manager to forcibly terminate the `ServletExecApache` application.

Remote Operation

By default `ServletExecApache` is installed on the same machine as the Apache Server and only accepts requests from the local machine (address 127.0.0.1). It's possible to install and configure the `ServletExecApache` application to run a different machine (the "remote" machine) than the Apache Server. To run `ServletExecApache` on a remote machine perform the following steps:

1. Add the following line to the Apache Server **httpd.conf** file:

```
ServletExecInstances <ip address>:<port>
```

where `<ip address>` is the IP address of the machine on which `ServletExecApache` will be running (the remote machine), and `<port>` is the port number on which `ServletExecApache` will communicate. Restart the Apache Server after modifying the **httpd.conf** file.

2. Copy the `ServletExec` directory and its contents to the remote machine.
3. Copy the Apache Server **mime.types** file to the `ServletExec` directory on the remote machine.
4. Edit the `ServletExecApache.bat` batch file on the remote machine and modify the `-root` parameter to specify the directory on the remote machine that will be used as the document root directory. Change the `-mimetypes` parameter to specify the path to the **mime.types** file on the remote machine. Finally, add the following argument to the end of the java command line:

```
-allow <ip address>
```

where `<ip address>` is the IP address on the machine on which the Apache Server is running.

Run the `ServletExecApache.bat` batch file on the remote machine to start the `ServletExecApache` application.

Multiple Java VMs

It's possible to run multiple instances of the `ServletExecApache` application, each with its own Java VM. In this configuration, each `ServletExecApache` instance is associated with a single, unique virtual host (domain). This is particularly useful for ISP configurations where it is desirable to isolate the servlets for each virtual host in a separate VM.

The multiple `ServletExecApache` instances may be run on the same (local) machine as the Apache server, on a remote machine, or a combination of local and remote machines (that is, some instances may run locally and some may run remotely).

Perform the following steps to run multiple instances of ServletExecApache with a single Apache web server:

1. Run the ServletExec installer for each ServletExecApache instance, selecting a unique installation directory for each instance.
2. Start each ServletExecApache instance on a unique port using the `-p` option to the `runapache` script.
3. For each `<VirtualHost>` directive in the Apache Server **httpd.conf** file, add the following directive:

`ServletExecInstances <ip address>:<port>`

where `<ip address>` is the IP address of the machine on which the ServletExecApache instance for that virtual server is running (127.0.0.1 for the local machine), and `<port>` is the port number on which ServletExecApache will communicate. Restart the Apache Server after modifying the **httpd.conf** file.

See the additional instructions for running ServletExecApache on a remote machine under the heading Remote Operation, above.

Uninstalling ServletExec

1. Use the Add/Remove Program control panel to remove ServletExec.
2. Undo the manual configuration steps described above. In particular, remove the entries added to the Apache Server **httpd.conf** configuration file.
3. Delete the ServletExec directory.



Appendix C2. Apache Installation & Operation (UNIX)

Completing and Verifying Your ServletExec Installation Starting and Stopping ServletExecApache

This appendix contains important information that will allow you to complete and verify your installation of ServletExec for Apache on UNIX, and contains instructions for starting and stopping the ServletExecApache application. It will also be useful if you decide to uninstall ServletExec to make sure you've completely removed all installed components.

In order to complete the ServletExec installation, you must manually modify one of the Apache configuration files. Instructions for modifying the Apache `srm.conf` file are provided under the heading Manual Configuration, below (note that `srm.conf` has been deprecated in favor of `httpd.conf` beginning with Apache 1.3.4). In addition, the ServletExec installation script automatically modified the Apache `httpd.conf` configuration file; a description of this change is also provided below.

ServletExec for Apache is implemented as an out-of-process application that communicates with the Apache Server via network sockets; a small native-code module is added to the Apache Server to manage this communication. This implementation allows ServletExec to run on a remote machine from the Apache Server. Instructions for running ServletExec on a remote machine are provided under the heading Remote Operation, below.

System Requirements

- ServletExec 2.2 for Apache has been tested on Solaris 2.5.1, 2.6, and 7 for SPARC and x86, on HP-UX 10.20, and on various versions of Linux. ServletExec for Apache should work on any UNIX variant that supports Apache 1.3.1 or higher and a JDK 1.1- or 1.2-compliant VM.
- ServletExec 2.2 supports the Apache Server version 1.3.1 and higher. The Apache server must be built with Dynamic Shared Object (DSO) support enabled. ***If you have previously built Apache without DSO enabled, you'll also need to rebuild Apache's***

apxs utility. Perform the following steps to rebuild Apache and **apxs** with DSO enabled:

```
$cd <apache source directory>
$rm src/support/apxs
$./configure --prefix=/usr/local/apache --enable-module=so
$make
$make install
```

- Before installing ServletExec you must first install a JDK 1.1- or 1.2-compliant VM. JDK or JRE 1.1.x for Solaris can be downloaded from Sun Microsystems' web site:

<http://java.sun.com/products/jdk/1.1/index.html>

<http://java.sun.com/products/jdk/1.1/jre/index.html>

JDK or JRE 1.2 for Solaris can be downloaded from:

<http://java.sun.com/products/jdk/1.2/index.html>

<http://java.sun.com/products/jdk/1.2/jre/index.html>

The JDK for HP-UX can be downloaded from Hewlett-Packard's web site:

http://www.internetsolutions.enterprise.hp.com/2_60_index.html

A JDK 1.1-compliant VM for Linux is available from:

<http://www.blackdown.org/java-linux.html>

Uninstall Other Servlet Engines

VERY IMPORTANT! You **must** uninstall any other servlet engines you may have previously installed for use with the Apache Server before installing and using ServletExec. In particular, modifications you may have made to the **httpd.conf** and **srm.conf** configuration file for other servlet engines must be removed.

Manual Configuration

In order to complete the ServletExec installation, you must manually modify the Apache **srm.conf** configuration file. In addition, the ServletExec installation script automatically modified the Apache **httpd.conf** configuration file. *You must stop and restart the Apache Server after modifying these configuration files.*

srm.conf

(Note: beginning with Apache 1.3.4, use of **srm.conf** has been deprecated in favor of **httpd.conf**. For Apache 1.3.4 and later, the following directives should be added to **httpd.conf**). Add the following lines to the end of the **srm.conf** file:

```
#
# ServletExec directives
#
```

```
<Location /servlet>
    SetHandler servlet-exec
</Location>

AddHandler servlet-exec jhtml
AddHandler servlet-exec jsp
AddHandler servlet-exec shtml
```

These directives define the default prefix and suffix aliases used by ServletExec (see Chapter 2 for a general discussion of servlet aliases, and further discussion of configuring prefix and suffix aliases with Apache, below).

NOTE: if you plan to use the Apache Server rather than ServletExec to parse server-side includes (SSI) pages using the file extension `.shtml` then don't add the last `AddHandler` directive to the **httpd.conf** file.

Prefix Aliases

For each prefix alias that is configured via the ServletExec Admin UI, a `Location` directive must be added to **httpd.conf** in the following format:

```
<Location /prefix-alias>
    SetHandler servlet-exec
</Location>
```

In the manual configuration instructions, above, ServletExec is configured to handle the `/servlet` prefix alias by default.

Suffix Aliases

There are two methods for configuring suffix aliases with Apache. The most common method is illustrated above. For each suffix alias that is configured via the ServletExec Admin UI, add an `AddHandler` directive to **httpd.conf** with the following format:

```
AddHandler servlet-exec <suffix-alias>
```

This method should be used if the URL Rewriting option of the Session Tracking feature is disabled, which it is by default (see Chapter 4 for a discussion of Session Tracking and URL Rewriting). This method of configuring suffix aliases provides better performance than the alternative method.

If the URL Rewriting option of the Session Tracking feature is enabled, an alternative method of configuring suffix aliases must be used. For this method, a `Location` directive is used instead of the `AddHandler` directive. For each suffix alias that is configured via the ServletExec Admin UI, add a `Location` directive to **httpd.conf** with the following format:

```
<Location /*.suffix-alias*>
    SetHandler servlet-exec
</Location>
```

Also, for each sub-directory level beneath the web server document root directory, an additional `Location` directive must be added to **httpd.conf**. For example, assume the web server root document directory is `/usr/local/apache/htdocs` and the following sub-directories appear beneath `htdocs`:

```
Htdocs/subdir1
Htdocs/subdir2
Htdocs/subdir2/subdir3
```

In this example, there are two levels of sub-directories beneath `htdocs`, so to create a suffix alias of `.jsp`, three `Location` directives needs to be added to **httpd.conf**, one for `htdocs` and one for each sub-directory level:

```
#
# suffix alias for htdocs
#
<Location /*.jsp*>
    SetHandler servlet-exec
</Location>

#
# suffix alias for htdocs/subdir1 and htdocs/subdir2
#
<Location /*/*.jsp*>
    SetHandler servlet-exec
</Location>

#
# suffix alias for htdocs/subdir2/subdir3
#
<Location /*/*/*.jsp*>
    SetHandler servlet-exec
</Location>
```

httpd.conf

The `ServletExec` installation script added the following directive to the **httpd.conf** file with the other `LoadModule` directives (near the top of the file):

```
LoadModule servletexec_module libexec/mod_servletexec.so
```

This directive causes the Apache Server to load the `ServletExec` native code module that manages communication with the `ServletExec` Apache Java application.

What Was Installed?

When you installed `ServletExec`, these changes were made to your system:

- the `servletexec` directory was created
- the file `mod_servletexec.so` was added to the Apache `libexec` directory

The servletexec Directory

The servletexec directory was created under `/usr/local`. The servletexec directory contains the License Agreement (in file `LICENSE`) and several sub-directories. The contents of these sub-directories are described below.

<code>bin</code>	contains the <code>runapache</code> script for starting the <code>ServletExecApache</code> application; this script is described in detail below
<code>doc</code>	contains this manual, a tutorial for writing servlets, and the Java Servlet API documentation from the Java Servlet Development Kit (JSDK) 2.1
<code>lib</code>	contains the <code>ServletExecApache.jar</code> file, a Java archive that contains the classes that implement the <code>ServletExecApache</code> application
<code>lib</code>	contains the <code>ServletExecApache.jar</code> , <code>servlet.jar</code> , and <code>ServletExec22.jar</code> files; the first archive contains the <code>ServletExecApache</code> application, the latter two archives contain classes that are required by the <code>ServletExecApache</code> application and are added to the <code>ServletExec</code> classpath in the <code>runapache</code> script
<code>src</code>	contains the file <code>mod_servletexec.c</code> that contains the source code for the <code>ServletExec</code> native module that manages communication between the Apache Server and the <code>ServletExecApache</code> application
<code>Examples</code>	contains examples of some of <code>ServletExec</code> 's advanced features, including: Server-Side Includes (SSI), Java Invoker (dynamic page compilation), JavaServer Pages (JSP), File Upload servlet, and Presentation Templates
<code>ServletLogs</code>	contains the <code>Servlet.log</code> files; entries are added to the <code>Servlet.log</code> file whenever a servlet invokes its <code>log()</code> method; see the section in Chapter 2 on configuring the servlet logging feature
<code>ServletExecData</code>	contains <code>ServletExec</code> configuration files; DO NOT MODIFY THE CONTENTS OF THIS DIRECTORY ; in emergency situations you may delete this entire directory and <code>ServletExec</code> will recreate it using its default configuration; however, if you do this all changes you've made to the default configuration will be lost and you will be required to re-enter your <code>ServletExec</code> serial number
<code>Servlets</code>	the default location for servlet class files; initially contains sample servlets that ship with <code>ServletExec</code>

Do not move the `servletexec` directory after installation.

ServletExecApache Operation

ServletExecApache is a standalone Java application that communicates with the Apache Server via network sockets. ServletExecApache is started using the `java` command on the command-line; in addition to `java` command arguments, ServletExecApache accepts several required and optional arguments.

The ServletExecApache application can be started using the `runapache` script, or directly from the command line using the `java` command.

runapache

The **runapache** script is provided for your convenience in starting the ServletExecApache application. **runapache** includes defaults for all of the `java` and ServletExecApache command-line arguments; edit `runapache` to modify these command-line arguments. A complete list of ServletExecApache command-line arguments is provided under the heading Manual Startup, below.

Manual Startup (java Command)

You can start the ServletExecApache application using the `java` command as follows:

```
java [java arguments] ServletExecApache [arguments]
```

The `java` command arguments must include the `-classpath` option specifying the path to the `ServletExecApache.jar` file.

The ServletExecApache arguments are:

```
-help  
-port <port number>  
-backlog <length>  
-home <ServletExecApache home directory>  
-root <path to server's document root directory>  
-root <path to the virtual server's document root directory>  
-mimetypes <path to Apache Server mime.types file>  
-allow <ip1,ip2,...,ipn>
```

If an option value contains spaces then it must be placed inside double quotes.

The `-port` argument is optional; the default value is 8888. This argument specifies the TCP/IP port on which the ServletExecApache application communicates with the Apache Server. If you modify this value you must also add the following variable to the Apache Server **httpd.conf** file:

```
ServletExecInstances <ip address>:<port>
```

where `<ip address>` is the IP address of the machine on which ServletExecApache is running (127.0.0.1 if ServletExecApache is running on the local machine; see the discus-

sion below for running ServletExecApache on a remote machine), and <port> is the port number on which ServletExecApache will communicate. Restart the Apache Server after modifying the **httpd.conf** file. Restart the Apache Server after modifying the **httpd.conf** file.

The `-backlog` argument is optional; the default value is 50. This argument specifies the size of the ServletExecApache incoming request queue.

The `-home` argument is optional; the default is the directory from which the `java` command is executed. This specifies the directory in which ServletExecApache will look for the ServletExec Data, Servlet Logs, and Servlets sub-directories.

The `-root` argument is required. It specifies the path to the Apache Server's document root directory. If ServletExecApache is being used with virtual servers, the document root directory of each virtual server must be configured using the `-root` argument.

The `-mimetypes` argument is optional. It specifies the path to the Apache Server **mime.types** file. If this argument isn't provided, the `ServletContext.getMimeType()` method always returns `null`.

The `-allow` parameter is optional. It specifies the address(es) of the Apache Server(s) that are allowed to communicate with the ServletExecApache application. An IP address can include the "*" character to indicate a subrange (for example: 168.121.97.*).

Stopping ServletExecApache

The ServletExecApache application must be stopped from the Shutdown page of the ServletExec Admin UI (see Chapter 2). If ServletExecApache is not shutdown properly, `destroy()` servlet methods are not invoked, SSI counters are not saved, sessions are not saved, and buffered log messages are lost.

Remote Operation

By default ServletExecApache is installed on the same machine as the Apache Server and only accepts requests from the local machine (address 127.0.0.1). It's possible to install and configure the ServletExecApache application to run a different machine (the "remote" machine) than the Apache Server. To run ServletExecApache on a remote machine perform the following steps:

5. Add the following line to the Apache Server **httpd.conf** file:

```
ServletExecInstances <ip address>:<port>
```

where <ip address> is the IP address of the machine on which ServletExecApache will be running (the remote machine), and <port> is the port number on which ServletExecApache will communicate. Restart the Apache Server after modifying the **httpd.conf** file.

1. Copy the `servlethexec` directory and its contents to the remote machine.

2. Copy the Apache Server **mime.types** file to the `servletexec` directory on the remote machine.
3. Edit the **runapache** script on the remote machine and modify the `-root` parameter to specify the directory on the remote machine that will be used as the document root directory. Change the `-mimetypes` parameter to specify the path to the **mime.types** file on the remote machine. Finally, add the following argument to the end of the java command line:

`-allow <ip address>`

where `<ip address>` is the IP address on the machine on which the Apache Server is running.

Run the **runapache** script on the remote machine to start the ServletExecApache application.

Multiple Java VMs

It's possible to run multiple instances of the ServletExecApache application, each with its own Java VM. In this configuration, each ServletExecApache instance is associated with a single, unique virtual host (domain). This is particularly useful for ISP configurations where it is desirable to isolate the servlets for each virtual host in a separate VM.

The multiple ServletExecApache instances may be run on the same (local) machine as the Apache server, on a remote machine, or a combination of local and remote machines (that is, some instances may run locally and some may run remotely).

Perform the following steps to run multiple instances of ServletExecApache with a single Apache web server:

1. Run the ServletExec installer for each ServletExecApache instance, selecting a unique installation directory for each instance.
2. Start each ServletExecApache instance on a unique port using the `-p` option to the **runapache** script.
3. For each `<VirtualHost>` directive in the Apache Server **httpd.conf** file, add the following directive:

`ServletExecInstances <ip address>:<port>`

where `<ip address>` is the IP address of the machine on which the ServletExecApache instance for that virtual server is running (127.0.0.1 for the local machine), and `<port>` is the port number on which ServletExecApache will communicate. Restart the Apache Server after modifying the **httpd.conf** file.

See the additional instructions for running ServletExecApache on a remote machine under the heading Remote Operation, above.

Uninstalling ServletExec

1. Undo the manual configuration steps described above. In particular, remove the entries added to the Apache Server **httpd.conf** and **srm.conf** configuration files.
2. Delete the `/usr/local/servletexec` directory.



Appendix D. Mac OS Installation

Verifying Your ServletExec Installation

This appendix contains important information that will allow you to verify your installation of ServletExec for Mac OS web servers. It will also be useful if you decide to uninstall ServletExec to make sure you've completely removed all installed components.

System Requirements

- ServletExec 2.2 requires a PowerPC-based Macintosh or Mac OS-compatible computer. ServletExec 2.2 *does not* run on 68K-based machines.
- ServletExec 2.2 requires Mac OS 8.1 or higher.
- ServletExec 2.2 requires Mac OS Runtime for Java (MRJ) version 2.0 or 2.1, which is included on the Mac OS 8.1 installation CD or can be downloaded from:

<http://www.apple.com/macos/java/text/download.html>

- ServletExec 2.2 *does not* work with MRJ 1.5 or 1.0.2.
- ServletExec 2.2 has been tested with the following web servers: WebSTAR 2.0, 2.1, and 3.0, Quid Pro Quo 1.0.2, 2.0, and 2.1, WebTen 1.1.1 and 2.0, and AppleShareIP 5.0.2 and 6.1. Here are the results of these tests:
 - ServletExec 2.2 works properly with WebSTAR 2.0, 2.1, and 3.0.
 - ServletExec 2.2 *does not* work with Quid Pro Quo 1.0.2.
 - ServletExec 2.2 works properly with Quid Pro Quo 2.0 and 2.1, however, both ServletExec and Quid Pro Quo use the .shtml suffix for server-side includes (SSI) files. Either Quid Pro Quo's suffix mapping or ServletExec's alias mapping for the SSIServlet needs to be changed to remove this conflict.
 - ServletExec 2.2 works properly with AppleShareIP 5.0.2 and 6.1, but it may be necessary to increase the plug-in memory allocation to 1024K. The MRJ just-in-time-compiler (JITC) is disabled when running ServletExec with AppleShareIP.
 - ServletExec 2.2 works properly with WebTen 1.1.1 and 2.0. The MRJ just-in-time-compiler (JITC) is disabled when running ServletExec with WebTen.

Uninstall Other Servlet Engines

VERY IMPORTANT! You **must** uninstall any other servlet engine you may have previously installed for use with your web server before installing and using ServletExec. In general, simply remove the other servlet engine from your web server's Plug-Ins folder.

(WebSTAR 3.0 users: WebSTAR 3.0 bundles a freeware servlet engine that must be removed from the Plug-Ins folder before using ServletExec. The name of this plug-in is "WebSTAR Java Servlet Runner". You'll also need to remove all `jrun*.zip` files from the System Folder:Extensions:MRJ Libraries:MRJClasses folder.).

What Was Installed?

When you installed ServletExec, the following items were added to your system:

- a folder named `ServletExec 2.2` containing documentation and example files demonstrating some of ServletExec's advanced features
- a folder named `Servlets` containing sample servlets was added to your web server's folder; this is the default location for servlet class files
- the ServletExec plug-in was added to the Plug-Ins folder within the web server's folder
- the `ServletExec.shlb` shared library was added to the Extensions folder within your System Folder

In addition to the above items, when ServletExec initializes the first time it creates a folder named `ServletExec Data` containing the ServletExec configuration files. **DO NOT MODIFY THE CONTENTS OF THIS FOLDER.** In emergency situations you may delete this entire folder and ServletExec will recreate it using its default configuration; however, if you do this all changes you've made to the default configuration will be lost and you will be required to re-enter your ServletExec serial number.

Uninstalling ServletExec

Perform the following steps to uninstall ServletExec for Mac OS web servers:

1. Delete the file `ServletExec.shlb` from the System Folder:Extensions folder.
2. Delete the ServletExec plug-in and `ServletExec Data` folder from within the web server's Plug-ins folder.
3. Delete the `ServletExec 2.2` folder and `Servlets` folder from within the web server's folder.



Appendix E. Developing Servlets & Miscellany

Tips for Developing Servlets Using ServletExec

The best way to develop servlets that you plan to deploy on ServletExec is to use the ServletExec Debugger 2.2. The ServletExec Debugger is a basic web server that has the complete ServletExec servlet engine built in. The ServletExec Debugger allows you to develop and debug your servlets in an environment that very closely simulates the ServletExec deployment environment.

The ServletExec Debugger can be used with almost any Java Integrated Development Environment (IDE) and comes with detailed instructions for use in the following IDEs:

- Symantec Visual Café 2.5 and 3.0
- Borland JBuilder 2 and 3
- Metrowerks CodeWarrior Pro 5
- IBM VisualAge for Java 2.0
- Sybase PowerJ 3.0

Support for additional IDEs is being added continuously; check our web site to see if support for your favorite IDE has been added. The ServletExec debugger can be downloaded from our web site:

<http://www.newatlanta.com/downloads/index.html>

Best of all, the ServletExec Debugger is absolutely **FREE** for all users!

ServletExec Tech Support FAQ

In addition to answering common questions about ServletExec installation and configuration problems, the on-line ServletExec technical support FAQ is constantly updated with debugging tips, bug reports, workarounds, and issues with interactions with third-party products:

<http://www.newatlanta.com/support-faq.html>

Java Servlet API Documentation

In order to develop servlets, you'll need the Java Servlet API documentation from the Java Servlet Development Kit (JSDK) 2.1. A copy of the Java Servlet API documentation is included in the ServletExec Documentation directory.

You can download the entire JSDK 2.1 from:

<http://java.sun.com/products/servlet/>

Also useful is the Java Web Server 1.1 documentation:

<http://jserv.javasoft.com/products/java-server/documentation/webserver1.1/index.html>

You may also be interested in a tutorial on developing servlets published online by Web Review magazine:

<http://webreview.com/97/10/10/feature/index.html>

Other valuable on-line resources include:

Servlet Central – an independent magazine dedicated to servlets and server-side Java programming. <http://www.servletcentral.com/>

Servlets.com – a site primarily dedicated to promoting Jason Hunter's book, "Java Servlet Programming" (O'Reilly & Associates, 1998), which is an excellent resource and highly recommended for all servlet programmers. <http://www.servlets.com/>

Servlet Threads

Normally, a servlet's `doGet()`, `doPost()`, and `service()` methods may be invoked concurrently on multiple threads. Therefore, these methods must be thread-safe. Here are some general guidelines to follow:

- Parameters to these methods and local variables declared within these methods are thread-safe; no special effort is required on your part.
- Class variables (those not defined within a method) are not thread-safe. Access to class variables must be protected by using Java's `synchronized` keyword.
- DO NOT declare your `doGet()`, `doPost()`, or `service()` methods to be `synchronized`. Doing so will allow only a single thread to execute through your servlet, which will seriously degrade performance. Instead, use `synchronized` code blocks or other `synchronized` methods to control access to your class variables.

If you don't want to worry about threading issues, you can declare your servlet classes to implement the `SingleThreadModel` interface. Doing so guarantees that no two threads

will execute concurrently in the servlet (see the Java Servlet API documentation). Because of potential performance degradation, use of the `SingleThreadModel` interface is not recommended.

For more detailed information regarding Java threads and use of the synchronized keyword, the following two references are highly recommended:

Concurrent Programming in Java

by Doug Lea

ISBN 0-201-69581-2

Java Threads

by Scott Oaks & Henry Wong

ISBN 1-56592-216-6

Debugging Servlets

The `ServletExec` Debugger should be your first choice for debugging servlets (see the discussion at the beginning of this appendix). However, sometimes it's necessary to debug your servlets running in a "live" deployment environment.

One of the tried-and-true debugging techniques is to include debug calls to `System.out` and `System.err` in your servlet code. For most web servers, the output from `System.out` and `System.err` is written to the `ServletExec.log` file. Also, all exceptions and errors caught by `ServletExec` are written to this log file.

Output to `System.out` and `System.err` is not buffered, but is written immediately to the `ServletExec.log` file. Therefore, you should remove extraneous calls to `System.out` and `System.err` from your code before going into production, otherwise your servlet won't perform as well as it could. Use the servlet `log()` method for routine output in your servlets. Output to the `log()` method is buffered by `ServletExec` and written to the `Servlet.log` file by a background thread, thereby minimizing impact on performance of your servlet.

For web servers on Windows NT, an application named DBMON (`dbmon.exe`) is provided that creates a console window to which `System.out` and `System.err` are redirected, and to which exceptions and errors caught by `ServletExec` are logged. Simply launch DBMON anytime `ServletExec` is running. (DBMON does not work on Windows 95/98). You can find DBMON within the `ServletExec` installation directory.

Profiling Servlets

There are several commercial Java profiling tools that can be used to profile servlets running within `ServletExec`. The tools are: `JProbe`, `OptimizeIt` and `NuMega DevPartner`.

Instructions for using these tools with ServletExec can be found in our on-line technical support FAQ:

<http://www.newatlanta.com/support-faq.html>

Native Methods

If your servlet needs to invoke native methods, the Java class that loads the native library (via `System.loadLibrary()`) cannot reside in the `Servlets` directory due to a JDK bug. To work around this bug, place the class that loads the native library in any directory other than the `Servlets` directory, then add the directory that contains this class to the ServletExec classpath (see Chapter 2 for instructions on modifying the ServletExec classpath).

Static (Class) Variables

If you have a Java class that contains static variables that must be accessed by multiple servlets, or multiple instances of the same servlet, do not place this class in the `Servlets` directory. Place the class that contains the static variables in any directory other than the `Servlets` directory, then add the directory that contains this class to the ServletExec classpath (see Chapter 2 for instructions on modifying the ServletExec classpath).

Shared Classes

Classes that are shared by multiple servlets, or multiple instances of the same servlet, cannot be placed in the `Servlets` directory. Instead, such shared classes must be placed in a directory that appears in the ServletExec classpath (see Chapter 2 for a discussion of the ServletExec classpath).

HttpSession Value Classes

When using the Session Tracking feature, servlet developers are free to store and retrieve objects of any class as values in the `HttpSession` object using the `putValue()` and `getValue()` methods. However, the class files for such classes must not be placed in the `Servlets` directory. Instead, they must be placed in a different directory, which must then be added to the ServletExec classpath (see Chapter 5 for a discussion of the Session Tracking feature; see Chapter 2 for discussions of the `Servlets` directory and the ServletExec classpath).

Servlet Names

The Java Web Server 1.1 supports two servlet methods that are not part of the Java Servlet API (that is, they are not defined by the `HttpServlet` class or any of its superclasses). These methods allow you to set and get the name of a servlet, where the “name” is either

the configured name or the class name if the servlet is not configured (see Chapter 2 on configuring servlets); these methods are also supported by ServletExec 2.2.

These methods have the following signatures:

```
public String getServletName();  
public void setServletName( String name );
```

When a servlet is instantiated, ServletExec introspects the servlet and invokes the `setServletName()` method with the name of the servlet. The implementation of this method should store the servlet name in a private instance variable so that the servlet name can be obtained by invoking the `getServletName()` method.

The `setServletName()` method is invoked before the servlet's `init()` method is called, so the name is available to the servlet during initialization.

(Compiled Code) in Stack Traces

If you're using a Java VM that includes a JITC (such as JDK 1.1.6 and higher, or MRJ 2.0 on Mac OS), then you'll often see the text `(compiled code)` instead of line numbers in stack traces such as those printed in `ServletExec.log` when exceptions are generated. During development you may want to disable the JITC in order to see line numbers.

To disable the JITC for JDK 1.1.6 or higher, use the VM Settings page of the ServletExec Admin UI (see Chapter 2). To disable the JITC for MRJ 2.0, remove the file "MRJ PPC JITC" from the folder "System Folder:Extensions:MRJ Libraries".

User Accounts for Microsoft IIS

Because ServletExec runs as part of the Microsoft IIS process, your servlets will run under different user accounts at different times. This will primarily affect their ability to read and write to the file system because access to the NT File System (NTFS) is based on the user account of the process. Here are the rules:

1. During normal request processing in your servlet's `service()`, `doGet()`, or `doPost()` method, your servlet will be running under the user account of the authenticated user, if the user had to enter a username and password to access your servlet. Otherwise your servlet will be running under the `IUSR_<server name>` account (the account under which IIS runs anonymous user requests).
2. If your servlet is configured to be loaded by ServletExec during initialization, your `init()` method will be executed under the `SYSTEM` account. Otherwise, if your servlet is loaded when it receives its first request, the rules for item #1, above, apply.

3. Normally, your servlet's `destroy()` method will run under the `SYSTEM` account when it is invoked due to a shutdown of `ServletExec`. However, if your servlet is reloaded due to a class file modification, the rules for item #1, above, apply.

UNIX File Descriptors

If you're running `ServletExec` on a UNIX systems that receives a very high traffic volume, you may begin to receive "too many files open" errors. This means the file descriptor limit of your system is being exceeded.

You can examine the current setting of the file descriptor limit using the `limit` command. Use the following `cs` shell command to increase the file descriptor limit to 256 (for example):

```
limit descriptors 256
```

Use the following `sh` or `ksh` shell command to increase the file descriptor limit to 256 (for example):

```
ulimit -n 256
```

Other shells will have similar commands to allow you to increase the file descriptor limit.