

JWinSvc™

User's Guide

February 2004



CSCARE Inc.

The information in this document is subject to change without notice. Companies, names, and data used in examples herein are fictions unless otherwise noted.

Copyright (c) 2002-2004 CSCare Inc. All rights reserved worldwide.

JWinSvc is the trademark of CSCare Inc. Java and all Java based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. Other product names and company names mentioned herein may be the trademarks or registered trademarks of their respective owners.

Table of Contents

TABLE OF CONTENTS	3
CONTACT INFORMATION.....	4
FEATURES	5
<i>The Advantages of JWinSvc</i>	5
<i>Principle of Operation</i>	6
LICENSING	7
PREREQUISITES	8
<i>System Requirements</i>	8
<i>Java Requirements</i>	8
INSTALLATION	9
UPGRADING	9
UNINSTALLATION	9
JWINSVC USAGE DESCRIPTION	10
<i>Starting JWinSvc</i>	10
<i>Startup Method</i>	13
<i>How to Shutdown a Java Application</i>	14
<i>Renaming JWinSvc.exe</i>	15
<i>How to Choose the Desired Java VM for JWinSvc</i>	15
<i>Service Recovery</i>	17
JBoss 3.0.4 AND 3.2.3 - HOW TO RUN IT AS SERVICE WITH JWINSVC	18
TOMCAT 3.2.4, 3.3.1A, 4.0.1, 4.1.18 AND 5.0.16 - HOW TO RUN IT AS SERVICE WITH JWINSVC	18
ORION 1.5.4 AND 2.0.2 - HOW TO RUN IT AS SERVICE WITH JWINSVC	19
OC4J - ORACLE9IAS CONTAINERS FOR J2EE - HOW TO RUN IT AS SERVICE WITH JWINSVC.....	19
TROUBLESHOOTING	19
VERSION HISTORY	20
INDEX	22

Contact Information

CSCare Inc. welcomes your comments and suggestions on the quality and usefulness of JWinSvc. You can contact us on jwinsvc@cscare.com, or

CSCare Inc.
2880 Zanker Road, Suite 203
San Jose
CA 95134

Toll-free: 1-866-783-0663
Phone: (408) 806-6267
Fax: (408) 904-5620

For more information about CSCare Inc., please check the CSCare Web site at <http://www.cscare.com>.

Features

JWinSvc is a Java wrapper designed for Windows NT/2000/XP/2003 which enables to run your Java applications as usual NT services. JWinSvc can be used to install and uninstall Java applications as well as to start and stop them. Moreover, you can force your Java applications to shut down smoothly, avoiding malfunction or crash.

For every Java application run by JWinSvc as NT service, you can rename the particular JWinSvc to distinguish the applications running at the same time.

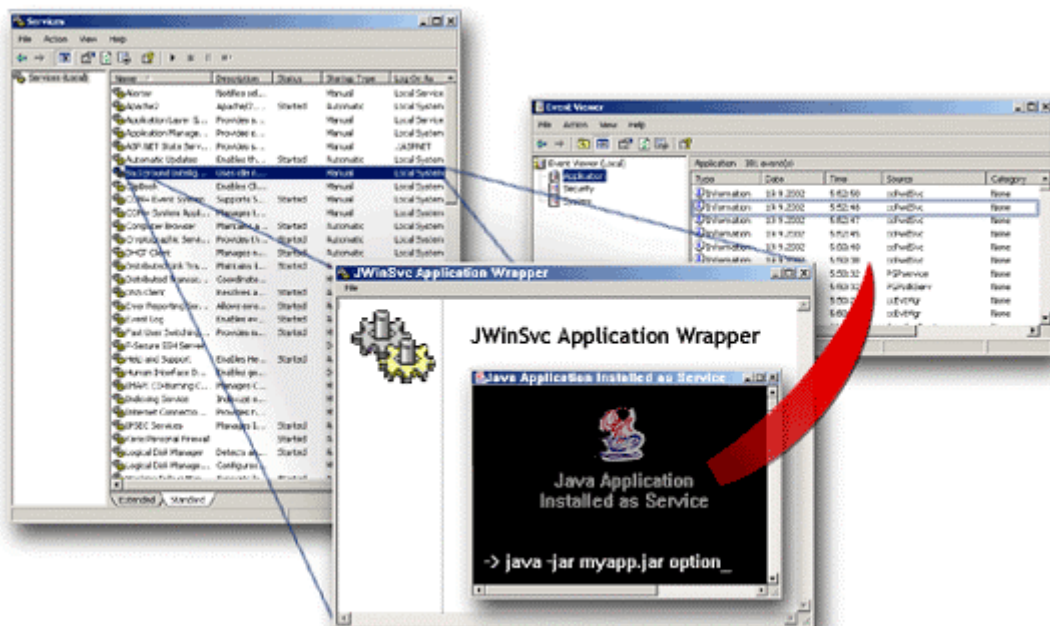
In addition, it is possible to set a working directory for your application, service applet name, and service applet description which appears in Task Manager and event logs.

The Advantages of JWinSvc

- enables to install and run Java applications as NT services on Windows NT, 2000, XP and 2003 Server platforms
- easily installs, starts, stops and uninstalls Java applications as services
- permits to specify service dependencies and allows services to interact with desktop
- provides several effective options to terminate Java applications smoothly when service stop is called, including a new unique method "System.exit() injection"
- supports Service Recovery in Service Control Manager for wrapped Java applications running with JWinSvc as services
- offers an option to specify a "startup method"
- enables to specify the preferred Java runtime library to be loaded when launching an application
- implemented as a sole exe file
- no need to re-compile or alter the installed Java applications
- supports JDK 1.2, 1.3 and 1.4, keeps the Java command line syntax and survives user logoff
- allows to rename itself in order to provide a better identification of the service in "Processes" in Task Manager
- application's System.out output displayed in Event Viewer can be optionally copied to a log file
- options to set the startup type of the installed service as well as the priority of the started service are implemented

Principle of Operation

The picture below shows the conceptual schema of JWinSvc operation:



- JWinSvc takes over the control of start and stop signals as well as signals to Event Viewer. The consequence is that the wrapped Java application acts as a usual NT service (having its name and description displayed in Task Manager and event logs, etc.).
- When the stop signal is received, JWinSvc tries to shutdown the Java application smoothly using the following approaches:
 1. JwinSvc uses "System.exit() injection" that calls System.exit() via JNI in a way as if the application itself called System.exit().
 2. If there is an external utility that takes care for stopping the Java application correctly, it can be called by specifying the `/shutdownCommand` command line parameter.
 3. In case that the Java application has a "shutdown" method implemented, it can be called by specifying the `/shutdown` command line parameter.

Shutdown methods and their usage with examples are described on the [How to Shutdown a Java Application](#) page.

Licensing

CSCare Inc. distributes the following three different versions of JWinSvc:

1. **Evaluation-only Edition** - allows for limited amount of time trying out whether the software fully meets the requirements. It permits to run only one service, does not allow changing the JWinSvc.exe file name in order to distinguish it as a module in Task Manager and writes evaluation remarks in logs. The started service stops after 24 hours.
2. **Personal Edition** - allows installing the Software on a single computer and running unlimited number of services on a single host. Renaming of JWinSvc.exe is possible. However, further resale and/or redistribution of the Software (e.g. with your software or services) to third parties is prohibited. No time limitations for running services.
3. **Enterprise Edition** - basic features are the same as of the Personal Edition. Redistribution of the Software as part of your software or services to third parties is allowed. An Annual Maintenance Plan is included.

What is covered by the **Annual Maintenance Plan**?

- Unlimited technical support on all products purchased for the term of the Plan.
- All upgrades to the products and modules originally purchased, which are officially released to production during the term of the Plan.
- All minor upgrades and patches which are officially released to production during the term of the Plan.
- Automatic e-mail notification when updates and/or upgrades become available.

Prerequisites

System Requirements

In order to run JWinSvc, one of the following operating systems must be installed:

- Windows NT 4.0
- Windows 2000
- Windows XP
- Windows 2003 Server

Java Requirements

Every compliant Java environment with JNI (Java Native Interface), as well as JDK and JRE 1.2, 1.3 and 1.4 are supported.

You can download the latest JavaSoft's JVM and Java Runtime Environment (JRE) from <http://java.sun.com/getjava>.

Installation

No installation is needed. Just copy the obtained JWinSvcInstall.exe file to any directory where you wish to place JWinSvc, and start it. JWinSvc Installer console window opens and you will be asked to type in the **unlock code** and press enter.

The unlock code is normally sent to you via e-mail to the address you entered during the JWinSvc download from our web site.

If you have not received the password, please contact us on jwinsvc@cscare.com.

The unlock code is case sensitive. After entering the right unlock code, JWinSvc.exe is extracted from the Installer file in the current directory and is ready to use.

Note: In order to **install/uninstall** services with JWinSvc, **administrator** rights are required; whereas to **start/stop** services, **Power User** rights are sufficient.

Upgrading

When upgrading JWinSvc to a newer version, or upgrading the evaluation edition of JWinSvc to a commercial edition, stop JWinSvc on the computer if running, and replace the older or evaluation version of the JWinSvc.exe file with the new exe file.

Uninstallation

To remove JWinSvc from your system, stop and uninstall all services installed and run by JWinSvc first, and then delete JWinSvc.exe from its directory. JWinSvc does not create additional files or write into other files or registry keys.

JWinSvc Usage Description

Starting JWinSvc

In order to start JWinSvc, open the command shell and type `jwinsvc` including the path (if applicable). Follow the usage displayed in the command shell to manage your Java applications.

Note: In order to **start/stop** services with JWinSvc, **Power User** rights are required. After moving the user to the Power Users group, system reboot is necessary for this change to take effect.

JWinSvc command line syntax looks as follows:

```
jwinsvc <ServiceName> {<command>} <Java command line>
```

The description of the items in bold are to be found below:

ServiceName

Service Name displays the name of the service applet and serves as the service unique identification, used to distinguish the service in the Service Control Manager (SCM) database. It also defines the service namespace in the registry; its registry key is created in `HKLM\SYSTEM\CurrentControlSet\Services` and in the Eventlog service Application database.

Commands

The following commands can be used and combined provided the sequence of chosen commands is possible to perform:

/help

displays usage hints

/install

registers the service with SCM (Service Control Manager), adds the service into the SCM database

/start

starts the service

/stop

stops the service

/uninstall

stops the service if running, and removes it from the SCM database

/cwd:<Directory>

sets the current working directory – this is the directory where from your Java application will be launched

/desc:<Description>

sets the service applet description

/depend:<ServiceName>

specifies the name of other service on which the service launched by JWinSvc depends. Multiple occurrences are allowed and the order of instances is preserved. The dependent services can be specified either by the **Service Name** or **Display Name** as defined in the 'Properties' dialog for the particular services in 'Control Panel/Administrative Tools/Services'.

Note: When dependency on groups is applied, the '+' prefix has to be added in front of the service group name, to distinguish it, as services and service groups use the same name space.

/interact

this parameter if present, tells JWinSvc to create a regular console window for the specified service to print the service's standard input and output streams there. Output from the wrapped application and

JWinSvc itself is logged in events and log files. This parameter has to be specified when installing the service, for example:

```
jwinsvc "Test Service" /install /interact java -jar test.jar
```

The console window is then created after the service is started. After installation, the interactive setting can be controlled directly via Service Properties, with the checkbox on the Log On tab. Changes applied here take effect after the service is restarted.

Note: A service launched by JWinSvc with the **/interact** option can be also stopped by closing the console manually, either with Ctrl-C key combination or window close. Logging off the user session also closes the service. The service now behaves as an application launched via Java command line.

/automatic

/manual

/disabled

using one of these three commands, the **startup type** of the installed service can be set. The same as setting the service properties in the Services applet.

/priority:<priority>

sets the priority of the installed service. The priority can be set to: low, normal, high or realtime.

Note: Priority can be set using the **/priority** parameter only during installation. Once set, it cannot be changed neither using the **/priority** parameter nor in Windows Task Manager.

If **/priority** parameter is not defined, then the service is started with the default priority 'normal'.

Example:

```
jwinsvc "My Service" /install /start /priority:low
```

In this case, the service is installed and started with low priority.

/startup[:<MethodName>]

main class method for application startup. As the startup method is prototyped as **public static boolean startup()**, the method name 'startup' can be overridden by specifying the <StartupMethod> argument.

/shutdown:<MethodName>

main class method for shutdown; this method will be called when you stop the service (from the service applet) or at system shutdown. If the Java application does not exit after a timeout (see the command **/timeout**), it will be terminated forcibly.

Shutdown method implemented in a Java application can look as follows:

```
public static void shutdown()
{
    closeSocket();
    disconnectDatabase();
    ...
}
```

For more detailed information, see the section [How to Shutdown a Java Application](#) on page 14.

/shutdownCommand:<command>

command line for application shutdown. As some existing Java applications provide other ways to invoke a proper shutdown sequence, e.g. running an external utility or command that makes use of interprocess communication, to control the application (TCP connection...). JWinSvc accommodates itself to these ways of shutting down an application and allows to specify a command line to perform in <command line> in the described command line parameter.

For more detailed information, see the section [How to Shutdown a Java Application](#) on page 14.

/timeout:<TimeLimit>

time limit for the Java application to start/stop. If not specified, then 30 seconds is used by default. The timeout value has to be specified in milliseconds. Set timeout to 10 seconds as follows:

```
/timeout:10000
```

/debug:<Level>

debugging level: 0=errors only, 1=errors with warnings, 2=detailed messages. If not specified, then (0) is set by default in order to log errors at least.

/logfile [:<Path>]

this parameter if present, tells JWinSvc to redirect logging of events into a local file named as '<ServiceName>_Event.log' or any other name specified by argument <FileName>. The log file is created if not found, in JWinSvc's working directory, and appended if exists. It has to be removed manually.

Format of log file records is as follows:

```
<date> <time> <severity>: <message>
```

where each new record begins on a new line.

Example:

```
jwinsvc "Test Service" /install /logfile:test.log java -jar test.jar
```

/quiet

disables logging, including Java application output, except for service operation

Example:

```
jwinsvc "Test Service" /install /quiet java -jar test.jar
```

Note: When logging into a local file or with logging disabled, NT Event logging is disabled except for messages concerning service operation (installation summary, records about service start and stop). Also note that with **/interact** parameter specified, the Java application output is not redirected and recorded in any logfile nor event log. The output, though, is not affected and goes into console window as expected.

/detach

launches the Java application in a separate process. Use it for Java applications that use System.exit which terminates the whole process including the Java VM. Then it will allow JWinSvc to notify the SCM of service exit and will cause timeout and error messages.

/runtime:lib:<path>

this parameter specifies the exact path to JVM library, for example:

```
/runtime:lib:"C:\jdk1.3.1\jre\bin\classic\jvm.dll"
```

Note: For detailed description of '/javahome' and '/runtime:lib' parameters, see the section [Command Line Parameters](#) on page 15.

/javahome:<directory>

this parameter specifies the path to Java installation base directory, for example:

```
/javahome:"C:\jdk1.3.1"
```

The JVM library will be searched in the 'C:\jdk1.3.1' directory including all its subdirectories.

Java Command Line

Java command line provides the same options as if 'java.exe' launcher was used, except the '-version', '-X', and '-help' options, which are not implemented.

Examples of Commands

A few examples are provided below in order to demonstrate the use of JWinSvc commands.

To install and start a service called 'My Service' from jar in its working directory called 'My Folder':

```
jwinsvc "My Service" /install /cwd:"My Folder" /start java
-jar myapplication.jar
```

To install a service called 'My Service', set its startup type to 'manual', set its priority to 'low' and start it:

```
jwinsvc "My Service" /install /manual /start /priority:low java
-jar myapplication.jar
```

Note: After restarting Windows, the service needs to be restarted manually.

To stop the service while running JWinSvc from the service's current directory:

```
jwinsvc "My Service" /stop
```

To start the service again:

```
jwinsvc "My Service" /start
```

To uninstall the service:

```
jwinsvc "My Service" /uninstall
```

To install and start another service from zip in its working directory:

```
jwinsvc "Other Services" /install /cwd:"Other Folder" /start
java -cp otherapplication.zip Main
```

To stop and uninstall 'Other Services' while running JWinSvc from the service's current directory:

```
jwinsvc "Other Services" /uninstall
```

To install and start 'Test Service' which depends on 'Remote Procedure Call (RPC)' and 'Workstation':

```
jwinsvc "Test Service" /install /depend:"Remote Procedure Call (RPC)"
/depend:Workstation java -jar test.jar
```

To install and start 'Test Service' and redirect event logs to the 'test.log' file.

```
jwinsvc "Test Service" /install /logfile:test.log java -jar test.jar
```

To install and start 'Test Service' using the 'shutdown.bat' file to terminate the application when service stop is requested.

```
jwinsvc "Test Service" /install
/shutdownCommand:"C:\test app\shutdown.bat" java -jar test.jar
```

Startup Method

JWinSvc reports that application has finished its initialization when JVM is created and main class is loaded, just before invoking main class' main method. That means that when the progress bar in Services applet fills up, it closes and the service status becomes "Started".

However, this is just the beginning when the Java application's initialization really starts. If this is an issue, there is an option to specify a main class' method that will be called repeatedly (in 1-second time period) to query application's startup status:

```
/startup[ :<MethodName> ]
```

JWinSvc, then, reports the application start later, when the return value from the startup method indicates a successful initialization.

As the startup method is prototyped as **public static boolean startup()**, the method name 'startup' can be overridden by specifying the <StartupMethod> argument.

JwinSvc returns the following boolean values during the startup process:

```
true ... application is starting  
false ... application startup is finished
```

How to Shutdown a Java Application

Generally, there is no way for Windows services manager to inform a Java program that it should stop. When the JWinSvc service command handler receives request to stop the service, it tries to shutdown a Java application gracefully and uses several approaches.

Using "System.exit() Injection"

After receiving the stop request, JWinSvc uses a unique method "System.exit() injection" that calls "System.exit()" via JNI in a way as if the Java application itself called "System.exit()". This is the smoothest way to terminate Java applications.

Using "/shutdownCommand" Parameter

As some existing Java applications use other ways to invoke a proper shutdown sequence, e.g. running an external utility or command that makes use of interprocess communication, to control the application (TCP connection...). JWinSvc accommodates itself to these ways of shutting down an application and allows to specify an external utility or command to be performed in <command line> in the described command line parameter.

Example:

```
jwinsvc "Test Service" /install /shutdownCommand:"C:\test  
app\shutdown.bat" java -jar test.jar
```

Note: Use CMD.EXE standard quoting to pass double quotes down to the shutdown command.

Example:

```
jwinsvc "Test Service" /install /shutdownCommand:"C:\test  
app\shutdown.bat \"parameter with quotes\" java -jar test.jar
```

Note: JWinSvc recognizes whether the entered command line specifies an executable or batch file.

Using "/shutdown" Parameter

In this case, JWinSvc solves the problem of application shutdown by implementing a JNI-based mechanism that calls a selected method of the Java application when service stop is requested.

JWinSvc uses JNI (Java Native Interface) to call the "shutdown" method in the main application class. Your main class (the class with the public static void main method) must implement a "public static void" shutdown method. The method can have any name but it must be "public static void" without parameters. It also must not throw any exceptions. Example of such method:

```
public static void shutdownRequested()  
{  
    ...  
}
```

JWinSvc calls this method once the user or system requests the service to stop. This happens during Windows shutdown and also when a user manually stops the service either from the Services Manager or via the "net stop" command. The method must force your application to stop as soon as possible, as by default, JWinSvc terminates the application after 30 seconds forcibly. If you know the shutdown process takes longer, you can set a different timeout with the /timeout command.

You must explicitly inform JWinSvc that your application implements a shutdown method and what its name is if different from "shutdown". This is done with the /shutdown command. If your shutdown method is implemented as above, is named as "shutdownRequested" and the shutdown process takes up to 40 seconds, the jwinsvc command will look as follows:

```
jwinsvc "My Service" /install /start /shutdown:shutdownRequested  
/timeout:40000 java -jar myjar
```

Note: JWinSvc attempts to invoke the "shutdown method" only if the /shutdown parameter is specified. Even if your Java application's main class contains the appropriate shutdown method with the specified signature `public static void shutdown()`, this method will be not called unless the /shutdown parameter is specified.

The shutdown method name is "shutdown" unless otherwise specified in the argument of the /shutdown parameter. Only the method name of the Java application's shutdown method can be modified, the method itself has to be public, static and without arguments and return value.

Renaming JWinSvc.exe

There is an option to rename JWinSvc.exe to distinguish it as a module in Task Manager if more JWinSvc-wrapped applications are running; (e.g. rename it as TrapConsole.exe). If using a uniquely renamed executable, the ServiceName parameter is optional (not needed) as the program will use the base of the module name (e.g. TrapConsole).

In case that the JWinSvc.exe is not renamed and there is requirement to distinguish the executable in Task Manager, the ServiceName parameter is required in order

Note: Renaming JWinSvc.exe is possible only in Personal and Enterprise Editions (see the section [Licensing](#) on page 7).

How to Choose the Desired Java VM for JWinSvc

JWinSvc NT Service Java wrapper launches Java applications in a Java Virtual Machine created via JNI interface. Runtime library `jvm.dll` implements this interface (`javai.dll` in older Java releases).

Generally, Java applications run with Java Runtime Environment's runtime library. However, there may be applications or services that require to run with Java Development Kit's runtime library as it provides some additional features in comparison with Java Runtime Environment's runtime library.

There are several ways how to specify the preferred runtime library to be loaded when launching an application.

Command Line Parameters

In JWinSvc, the following two command line parameters are available, using which the runtime library can be specified:

1. **/runtimelib:<path>** – contains the path to runtime library

Example:

```
JWinSvc "My Service" /start  
/runtimelib:"C:\jdk1.3.1\jre\bin\classic\jvm.dll"
```

With 'runtimelib' parameter, 'My Service' is started and `jvm.dll` is loaded from the specified location. If the runtime library is not found on the defined path, an error occurs stating that the specified runtime library cannot be found.

2. **/javahome:<path>** – contains the path to Java installation base directory

Example:

```
JWinSvc "My Service" /start /javahome:"C:\jdk1.3.1"
```

In this case, 'My Service' application is started and the first runtime library found in the 'C:\jdk1.3.1' directory (including its subdirectories) will be loaded. This command line parameter is more general and gives a wider range in terms of searching for runtime libraries.

The type of the runtime library can be specified if necessary, using the <jvm type> command line switch (-classic, -server, -hotspot; it will be described later) as follows:

```
JWinSvc "My Service" /start /javahome:"C:\jdk1.3.1" java -server
```

Here, 'My Service' application is started using the 'server' type of jvm.dll runtime library found within the scope of the 'C:\jdk1.3.1' directory. If no such runtime library is found, then an error occurs stating that the preferred ('server' type in this case) runtime library is not available.

How is JWinSvc Searching for Runtime Library

While launching an application with JWinSvc, it is searching for a runtime library in the following order:

1. If the command line parameter **/runtimeLib:<path>** is specified, JWinSvc will try to load the defined runtime library. If the runtime library is not found, you will be informed about it by an error message.
2. If the command line parameter **/javahome:<path>** is specified, JWinSvc will try to load a runtime library on the defined path extended by the **<relative path>** parameter which further specifies the locations where the runtime library may be located.

The **<relative path>** parameter is searched in the following order:

- \bin\<jvm_type>jvm.dll
- \bin\javai.dll
- \jre\bin\<jvm_type>jvm.dll
- \jre\bin\javai.dll

<jvm_type> parameter specifies the supported Java byte code interpreters given by Java installation and are listed in the **jvm.cfg** file located in Java home directory's **lib** or **lib\i386** subdirectory. These interpreters differ in terms of usage optimization.

There are three types of Java interpreters: 'classic' – the most stable but slightly slower interpreter than the next ones; 'server' – designed for server applications; and 'hotspot' – designed for applets.

JWinSvc takes the first jvm type in the order as listed in jvm.cfg.

Note: If you prefer a different jvm type to be selected, you can specify a Java option when launching your application, as for example:

```
JWinSvc "My Service" /install java -server -jar ap.jar
```

where the '-server' option can be also '-classic' or '-hotspot'.

If no runtime library is found on the defined path, you will be informed about it by an error message.

Note: In case of specifying the both command line parameters at the same time, the /runtimeLib parameter has higher priority, therefore the /javahome parameter will be ignored.

3. If no command line parameter is specified, JWinSvc will try to search for **<JAVA_HOME environment variable>** which specifies the path to the Java home directory. If it exists, JWinSvc performs the same way as if the '/javahome' command line parameter was used.

If the JAVA_HOME environment variable is set but no runtime library can be found on the defined path, an error message will inform you about it.

In case of not having the JAVA_HOME environment set, JWinSvc continues searching in locations in the order as follow:

4. **<current working directory><relative path>** – JWinSvc will be searching for a runtime library in the current working directory extended by the relative paths as described earlier. If no runtime library is found, JwinSvc continues searching in the following item:
5. **'HKEY_LOCAL_MACHINE\Software\JavaSoft\Java Development Kit'** registry key – JWinSvc checks the 'CurrenVersion' value, according to this value JWinSvc goes to the appropriate key (e.g. 1.3), and in that key, it takes the path defined in the 'RuntimeLib' value and performs the

same way as if the '/runtimelib' command line parameter was used. If no runtime library is found on the defined path, the following item will be searched:

6. **'HKEY_LOCAL_MACHINE\Software\JavaSoft\Java Development Kit'** registry key – 'CurrenVersion' value will be checked, according to this value JWinSvc goes to the appropriate key (e.g. 1.3), and in that key, it takes the path defined in the 'JavaHome' value and performs the same way as if the '/javahome' command line parameter was used. If no runtime library is found on the defined paths, the following item will be searched:
7. **'HKEY_LOCAL_MACHINE\Software\JavaSoft\Java Runtime Environment'** registry key – JWinSvc checks the 'CurrenVersion' value, according to this value, JWinSvc goes to the appropriate key (e.g. 1.3), and in that key, it takes the path defined in the 'RuntimeLib' value and performs the same way as if the '/runtimelib' command line parameter was used. If no runtime library is found on the defined path, the following item will be searched:
8. **'HKEY_LOCAL_MACHINE\Software\JavaSoft\Java Runtime Environment'** registry key – 'CurrenVersion' value will be checked, according to this value JWinSvc goes to the appropriate key (e.g. 1.3), and in that key, it takes the path defined in the 'JavaHome' value and performs the same way as if the '/javahome' command line parameter was used.

If no runtime library is found after all, JWinSvc will report an error stating that no runtime library is available on the system.

Service Recovery

Failure of a JWinSvc wrapped Java application affects JWinSvc as a service. The Service Recovery options (available in service Properties in the Services applet) take effect, however only under the circumstances described further.

JWinSvc wrapper tries to recognize whether the wrapped Java application failed. Java application failure is not clearly defined in general though. The main method in Java applications is not defined to return an exit code. Exceptions affect Java threads and JWinSvc monitors only the main, starting thread.

JWinSvc simulates the Java application failure to Service Control Manager when it detects one of the following conditions:

1. System.exit() method is called with a nonzero argument. According to Java API specification, by convention, it indicates an abnormal termination.
2. Exception is thrown out of the main method (when it is not handled).

JWinSvc will not consider those cases as a failure when a non-main thread fails and some critical functionality is lost. For example, when an application thread handling client request, listening on some TCP port, fails. The application 'failed' but not to JWinSvc, which cannot detect a problem definitely.

Service Recovery options, i.e. actions taken by Service Control Manager after it detects a service failure, are not settable by JWinSvc parameters. Use the Services applet to set these options.

JBoss 3.0.4 and 3.2.3 - how to run it as service with JWinSvc

JBoss is a Java application server. It is available from <http://www.jboss.org>.

In order to install JBoss as service with JWinSvc, please follow these steps:

1. Copy JWinSvc.exe into the "bin" subdirectory of the JBoss root directory (presumably C:\JBoss\bin).
2. Then download a zip file for JBoss version **3.0.4** from <http://www.cscare.com/JWinSvc/download/jboss-3.0.4.zip> and for version **3.2.3** from <http://www.cscare.com/JWinSvc/download/jboss-3.2.3.zip>. These files contain the necessary batch files needed to install, run and eventually uninstall JBoss as NT service with JWinSvc.
3. Unzip the file and copy the content of the "bin" folder into JBoss root directory's "bin" folder.
4. Finally, run "installsvc.bat" to install JBoss as NT service.
5. To uninstall JBoss as service with JWinSvc, run "uninstallsvc.bat".

Note: In order to change the service name, description or logging, please edit variables at the top of the "jbossvc.bat" file (see the comments inside).

Tomcat 3.2.4, 3.3.1a, 4.0.1, 4.1.18 and 5.0.16 - how to run it as service with JWinSvc

Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and JavaServer Pages technologies. It is available from <http://jakarta.apache.org/tomcat/>.

In order to install Tomcat as service with JWinSvc, please follow these steps:

1. Copy JWinSvc.exe into the "bin" subdirectory of the Tomcat root directory (e.g. C:\Jakarta-Tomcat-3.3.1a\bin).
2. Then download a zip file for the following Tomcat versions:
3.2.4 from <http://www.cscare.com/JWinSvc/download/jakarta-tomcat-3.2.4.zip>,
3.3.1a from <http://www.cscare.com/JWinSvc/download/jakarta-tomcat-3.3.1a.zip>,
4.0.1 from <http://www.cscare.com/JWinSvc/download/jakarta-tomcat-4.0.1.zip>,
4.1.18 from <http://www.cscare.com/JWinSvc/download/jakarta-tomcat-4.1.18.zip> and
5.0.16 from <http://www.cscare.com/JWinSvc/download/jakarta-tomcat-5.0.16.zip>.
These files contain the necessary batch files needed to install, run and eventually uninstall Tomcat as NT service with JWinSvc.
3. Unzip the file and copy the content of the "bin" folder into Tomcat root directory's "bin" folder.
4. JAVA_HOME environment variable is required to be set. Start the Command Shell and type "set". The set environment variables will be displayed in the console window. If JAVA_HOME is not present, add it by typing

```
set JAVA_HOME="<path to JRE or JDK on your system>"
```


e.g.

```
set JAVA_HOME="C:\jdk1.4.1_01"
```
5. Finally, run "installsvc.bat" to install Tomcat as NT service.
6. To uninstall Tomcat as service with JWinService, run "uninstallsvc.bat".

Note: In order to change the service name, description or logging, please edit variables at the top of the "tomcatvc.bat" file (see the comments inside).

Orion 1.5.4 and 2.0.2 - how to run it as service with JWinSvc

Orion is yet another Java application server. It is available from <http://www.orionserver.com>.

In order to install Orion as service with JWinSvc, please follow these steps:

1. Copy JWinSvc.exe into the "bin" subdirectory of the Orion root directory (presumably C:\Orion\bin).
2. Download a zip file for Orion version **1.5.4** from <http://www.cscare.com/JWinSvc/download/orion-1.5.4.zip> and for version **2.0.2** from <http://www.cscare.com/JWinSvc/download/orion-2.0.2.zip>. These files contain the necessary batch files needed to install, run and eventually uninstall Orion as NT service with JWinSvc.
3. Unzip the file and copy the content of the "bin" folder into Orion root directory's "bin" folder.
4. JAVA_HOME environment variable is required to be set. Start Command Shell and type "set". The set environment variables will be displayed in the console window. If JAVA_HOME is not present, add it by typing

```
set JAVA_HOME="<path to JRE or JDK on your system>"
```

```
e.g. set JAVA_HOME="C:\jdk1.4.1_01"
```

5. Finally, run "installsvc.bat" to install Orion as service.
6. To uninstall Orion as NT service with JWinSvc, run "uninstallsvc.bat".

Note: In order to change the service name, description or logging, please edit variables at the top of the "orionsvc.bat" file (see the comments inside).

OC4J - Oracle9iAS Containers for J2EE - how to run it as service with JWinSvc

Oracle9iAS Containers for J2EE (OC4J) is an application server available from <http://otn.oracle.com/tech/java/oc4j/>.

In order to install OC4J as service with JWinSvc, please follow these steps:

1. Copy JWinSvc.exe into the "\J2ee\home" subdirectory of the OC4J root directory (e.g. C:\oc4j\j2ee\home).
2. OC4J can be then installed as NT service with JWinSvc by running the following command from the "\j2ee\home" subdirectory.

```
jwinsvc OC4J /install /start java -jar oc4j.jar
```

3. To uninstall OC4J as NT service with JWinSvc, run the following command from the "\j2ee\home" subdirectory:

```
jwinsvc OC4J /uninstall
```

JWinSvc 1.3c has been tested with OC4J versions 9.0.3 and 9.0.4.

Troubleshooting

If you have any problems with running JWinSvc, please contact CSCare Inc. via e-mail on jwinsvc@cscare.com.

Version History

Version 1.3c

January 20, 2004

The following new features and bug fix are introduced in this release:

- Command line parameters `"/automatic"`, `"/manual"` and `"/disabled"` can set the startup type of the installed service
- Command `"/priority"` allows to set the priority of the started service
- Incorrect shutdown of a service in `"/interact"` mode after logoff is fixed
- Service Recovery in Service Control Manager for wrapped Java applications running with JWinSvc as services is supported

Version 1.3b

May 15, 2003

This update contains the following minor bug fixes:

- Exception logging and error reporting solved
- Problems when launching a main class which name consists of a compound name (e.g. `YourPackage.Main`) fixed
- Incorrect shutdown on Windows NT 4 operating system repaired
- Renaming of the `JWinSvc.exe` executable resolved

Version 1.3a

March 26, 2003

It solves the **rights management** issues when using JWinSvc. In order to install and uninstall services with JWinSvc, administrator rights are required, whereas to start and stop services, Power User rights are sufficient.

Version 1.3

March 05, 2003

It includes the following new features:

- **"System.exit() injection"** - a new unique method how to terminate a Java application correctly is introduced.
- Shutdown command - a new command line parameter `/shutdownCommand` is implemented to enable stopping Java application using scripts, batch files or external utilities.
- Startup command - allows controlling the start up process using a startup method implemented in the Java application.
- Alternative log file - application's `System.out` output displayed in Event Viewer can be optionally copied to a log file.
- Support for JDK and JRE 1.2 is added.
- Command line parameter `/quiet` serves as an option to disable logging, including the Java application output.

Version 1.2

November 15, 2002

This version enables to specify service dependencies and allows services to interact with desktop.

Version 1.1c

October 30, 2002

This update adds features for choosing the right Java VM from the ones installed on the computer.

Version 1.1b

October 24, 2002

It solves the problem with parameter parsing under special circumstances.

Version 1.1

September 18, 2002

Release of the first public version of JWinSvc.

Index

- C**
- Command 12, 15, 18, 19
 - Command line parameters 6, 11, 14, 15, 16, 17
 - Commands
 - automatic 11
 - cwd 10
 - debug 12
 - depend 10
 - desc 10
 - detach 12
 - disabled 11
 - help 10
 - install 10
 - interact 10
 - javahome 12
 - logfile 12
 - manual 11
 - priority 11
 - quiet 12
 - runtimeLib 12
 - shutdown 11
 - shutdownCommand 11
 - start 10
 - startup 11
 - stop 10
 - timeout 11
 - uninstall 10
- E**
- Event Viewer 5, 6
- J**
- Java command line 5, 10, 11, 12
 - Java wrapper 5, 15
- JAVA_HOME** environment variable 16, 18, 19
- javahome 12, 15, 16, 17
 - JBoss 18
 - jvm type 16
 - jvm.dll 12, 15, 16
- L**
- Licensing 7, 15
- N**
- NT services 5
- O**
- Orion 19
- R**
- Registry key 9, 10, 16, 17
 - Relative path 16
 - Runtime library 15
 - runtimeLib 12, 15, 16, 17
- S**
- Service applet 5, 10, 11
 - Service recovery 17
 - ServiceName 10, 12, 15
 - Shutdown 6, 11, 14
 - System.exit() injection 5, 6, 14
- T**
- Tomcat 18
- U**
- Unlock code 9
 - Upgrading to commercial version 9