



LifeKeeper for Windows v5.0

Script Porting Guide

August 2006

The product described in this book is a licensed product of SteelEye Technology, Inc.

SteelEye Technology and LifeKeeper are registered trademarks and SteelEye is a trademark of SteelEye Technology, Inc.

Microsoft, Windows, Windows 2000, and Windows 2003 are registered trademarks of Microsoft in the U.S. and other countries.

Cygwin™ DLL and utilities are Copyright © 2000, 2001, 2002, 2003, 2004, 2005 Red Hat, Inc.

Other brand and product names used herein are for identification purposes only and may be trademarks of their respective companies.

It is the policy of SteelEye Technology, Inc. to improve products as new technology, components, software, and firmware become available. SteelEye Technology, Inc., therefore, reserves the right to change specifications without prior notice.

To maintain the quality of our publications, we welcome your comments on the accuracy, clarity, organization, and value of this book.

Email correspondence to:

ip@steeleye.com

Copyright © 2006
By SteelEye Technology, Inc.
Palo Alto, CA U.S.A.
All Rights Reserved Worldwide.

| | |
|---|---|
| Introduction..... | 4 |
| Background: The Cygwin environment..... | 4 |
| File Access and the single-rooted filesystem..... | 4 |
| Mount Points..... | 5 |
| Script Modifications..... | 5 |
| PATH specification – colon-separated | 5 |
| LifeKeeper PATH variable | 5 |
| Replace ntps.exe | 6 |
| Replace registry.exe..... | 6 |
| Replace the “line” utility..... | 7 |
| The ‘let’ built-in command | 7 |
| tr -s ‘\’ | 7 |
| Environment variable enclosure is apostrophe, not quotation | 7 |
| Semicolons after “then” and “else” not allowed..... | 8 |
| awk match(..., “[<chars>]”) does not need \ before chars..... | 8 |
| “sh <shellname>” PATH search not performed without ‘-c’ | 8 |
| echo.exe does not translate character sequences..... | 8 |
| Filename case sensitivity with wildcard characters | 8 |
| Negative exit value from executable..... | 9 |
| Negative exit value from shell script | 9 |
| No automatic .ksh or .sh extension on argument to “sh” command..... | 9 |

LifeKeeper for Windows v5.0

Script Porting Guide

Introduction

SteelEye LifeKeeper for Windows v5.0 introduces a new Unix emulation layer environment based on the open source Cygwin™ project. This new environment provides nearly 100% compatibility with shell scripts developed for previous versions of LifeKeeper, but there are some differences which may impact existing custom scripts and/or any core LifeKeeper scripts that have been modified for specific customer requirements. All modified and custom LifeKeeper scripts should be investigated for compatibility. All LifeKeeper scripts shipped with the LifeKeeper for Windows and all LifeKeeper Recovery Kits have been certified for use with LifeKeeper for Windows version 5.0 and later.

Background: The Cygwin environment

Cygwin emulates a Posix environment within Windows. The manner in which Cygwin does this presents a few incompatibilities with the shell environment that was part of previous releases of LifeKeeper for Windows. These incompatibilities will be described here.

File Access and the single-rooted filesystem

Cygwin implements a single-rooted view of the host system's filesystem. Thus, all files can be accessed by a pathname starting with the "/" character. Files are accessible in two ways within Cygwin – by the familiar "<DRV>:/<path>", or by "/cygdrive/<DRV>/<path>". Since Cygwin allows the <DRV>:/<path> name format, most scripts do not have to change to accommodate Cygwin. One subtle difference that may affect a script is the fact that Cygwin does not keep track of the working directory on individual drive letters, and does not allow switching between drives by simply typing the drive letter. For instance, on a system with a D: and E: drive, in previous versions of LifeKeeper, a shell script could do the following:

```
# cd D:/tmp
# cd e:/e_dir
# cd D:
# pwd
D:/tmp
# cp file E:
# cd E:
# pwd
E:/e_dir
# ls
file
```

As you can see, previous versions of LifeKeeper act like a DOS prompt; they maintain the working directory on each drive. This is not the case in Cygwin: the full pathname would need to be provided each time the working directory was changed from one drive to another. This may have an impact on some LifeKeeper scripts.

Mount Points

Cygwin allows directories to be mounted at certain points in the filesystem. In LifeKeeper, the following mount points are configured:

| <u>Mount Point</u> | <u>Directory where mounted</u> |
|--------------------|--------------------------------|
| / | \$LKROOT/cygwin |
| /bin | \$LKROOT/bin |
| /usr/bin | \$LKROOT/bin |

So, in a bash shell, “ls /” will show the files and directories in \$LKROOT. “cd /etc” will actually take you to \$LKROOT/cygwin/etc. “/usr/bin/ls” will execute \$LKROOT/bin/ls.exe. To get to a location outside the \$LKROOT directory, you can use <DRV>:<path> or /cygdrive/<Drv>/<path> as described above.

Script Modifications

This section describes in detail what changes need to be made to custom and/or modified LifeKeeper scripts in order to make them operate correctly with LifeKeeper version v5.0.

PATH specification – colon-separated

As described earlier, Cygwin uses a single-rooted filesystem, with all files on a system being in paths that start at "/". Each drive letter is mapped to "/cygdrive/<DRV>". The PATH environment variable is constructed of these style paths, separated by colons. So, the following is NOT a valid PATH specification:

```
PATH=C:/lk/bin;$PATH          # INVALID
```

The correct way to specify this path modification would be:

```
PATH=/cygdrive/c/lk/bin:$PATH # VALID
```

When a shell starts, the Windows path is automatically converted to Cygwin format. If you need to add elements to the path, the "cygpath.exe" utility should be used. For instance, to add \$LKROOT/bin to the start of the path, use this:

```
PATH=`$LKROOT/bin/cygpath $LKROOT/bin`: $PATH
```

Since we set Cygwin mount points for /bin and /usr/bin to point to \$LKROOT/bin, the above call will put /usr/bin at the start of the PATH.

The same syntax can be used to add any path element to PATH. Just make sure to use \$LKROOT/bin/cygpath (or /bin/cygpath or /usr/bin/cygpath - they all are the same as explained in the section “Mount Points” above) to convert to the Cygwin format.

LifeKeeper PATH variable

In LifeKeeper v5.0, the PATH used by LifeKeeper processes does not include the entire system PATH. It has been limited to %SystemRoot%, %System32%, and %System32%\WBEM. If your recovery kit script depends on a path that includes directories other than these, it should set the PATH environment variable explicitly to include the directories that are needed.

Replace ntps.exe

The ntps utility is no longer provided in LifeKeeper for Windows. When given the "-a" option, ntps lists the entire command line of each running process.

This functionality is now provided in the LifeKeeper "ps" utility. If you give the "-f" option, the entire command line of each running process will be displayed.

Replace registry.exe

The registry utility is no longer provided in LifeKeeper for Windows. There are a number of alternative utilities that are provided, which can be used to replace registry.exe as described here.

The regtool.exe utility is now part of LifeKeeper, and can be used to replace registry.exe in many cases. One note: when specifying a path to a key using regtool, the path must begin with a "\" character. registry.exe, and the other LifeKeeper registry utilities described in the following paragraphs, do not work with a leading "\".

There are also a set of utilities that have always been part of LifeKeeper: CreateKey.exe, DeleteKey.exe, DeleteKeyValue.exe, GetSubKey.exe, GetValues.exe, QueryKey.exe, and SetKeyValue.exe. In all of these utilities, if a given registry key or value does not exist, an error is logged in the Application Log, so care should be taken to make sure the key/value exists before using these utilities to access it.

The following table gives the recommended syntax to use when replacing registry.exe in a shell script.

| | Pre-v5.0 | Post-v5.0 |
|-------------------------------|---|--|
| Test for existence of a key | registry -p -k <key> (then check \$?) | regdmp <key> or regtool check <key> (then check \$?) |
| Test for existence of a value | registry -p -k <key> -n <value> (then check \$?) | GetValues <key> grep "^<value>\$" or regtool -l list <key> grep "^<value>\$" (then check \$?) |
| Get a value | registry -p -k <key> -n <value> | QueryKey <key> <value> or regtool get "<key>\<value>" |
| Create / Set a value | registry -s -k <key> -n <value> -v <data> | SetKeyValue <key> <value> [REG_DWORD REG_SZ] <data> or regtool [-s -i -e -m] set "<key>\<value>" <data> |
| Delete a key | registry -d -k <key> | DeleteKey <key> or regtool delete <key> |
| Delete a value | registry -d -k <key> -v <value> | DeleteKeyValue <key> <value> or regtool unset <key>\<value> |

Note: there is no way to create a key using registry.exe. Use “CreateKey <key>” or “regtool create <key>”.

Replace the “line” utility

The utility “line.exe” is not provided in LifeKeeper v5.0. In previous versions of LifeKeeper, this utility could be used to print the first line of multi-line input.

In LifeKeeper v5.0, replace this with the following command:

```
head -1
```

The ‘let’ built-in command

The new LifeKeeper v5.0 ksh uses a slightly different syntax for the “let” command when performing arithmetic operations than previous versions of LifeKeeper. Rather than enclosing the arithmetic operation in `$(...<operation>...)`, LifeKeeper v5.0’s ksh uses `$((...<operation>...))`. So the command

```
let BITMASK=${BITMASK & 254}
```

should now be expressed as

```
let BITMASK=$((BITMASK & 254))
```

tr -s ‘\’

Previous versions of LifeKeeper included a tr.exe utility which could accept a single backslash argument as a substitution string. The tr shipped with LifeKeeper v5.0 requires two backslashes in order for it to recognize the input as a backslash. In most cases, shell scripts need to pass **four** backslashes to tr when it is executed inside backticks (the shell evaluates four down to two, and passes these to tr). For example – the command

```
CMD=`echo $DOSPATH | tr -s '\\' '/'`
```

should be expressed as

```
CMD=`echo $DOSPATH | tr -s '\\\\\' '/'`
```

Environment variable enclosure is apostrophe, not quotation

The LifeKeeper v5.0 ksh adds apostrophes around the values of environment variables which contain one or more spaces. In previous versions of LifeKeeper, ksh enclosed such values in quotation marks.

If you have code that looks for quotation marks around environment variable values, this should be changed to look for apostrophes. For instance, the following code in the SteelEye Data Replication restore.ksh script was used to remove quotations from the ExtMirrBase environment variable:

```
EMBASE=`echo $ExtMirrBase | cut -f2 -d"\"`
```

In LifeKeeper v5.0, this command is now

```
EMBASE=`echo $ExtMirrBase | cut -f2 -d\'`
```

Semicolons after “then” and “else” not allowed

The LifeKeeper v5.0 ksh is more strict in syntax checking of if-then-else clauses, and does not allow semicolons after then or else. The following command would have been valid in previous versions of LifeKeeper:

```
if [ $var = 1 ]; then; echo yes; else; echo no; fi;
```

but now should be expressed as

```
if [ $var = 1 ]; then echo yes; else echo no; fi;
```

awk match(..., “[<chars>]”) does not need \ before chars

The LifeKeeper v5.0 awk performs different syntax checking than the awk shipped with previous versions of LifeKeeper. If a script uses the awk “match()” function to check for the existence of certain characters in a string, then in most cases there is no need for a backslash before the characters. One exception to this is the backslash character itself – this requires **four** backslashes in most cases. The ‘]’ character requires two backslashes before it. And a quotation mark (“) should be preceded by one backslash.

“sh <shellname>” PATH search not performed without ‘-c’

The LifeKeeper v5.0 ksh does not automatically search its path when invoked without the ‘-c’ option. Also, the “.sh” and “.ksh” suffixes are not automatically appended without the ‘-c’ option. Thus, a command like:

```
$LKROOT/bin/sh getlocks args...
```

will not work as it did in previous versions of LifeKeeper. The preceding command should be changed to one of the following:

```
$LKROOT/bin/sh -c "getlocks args..."
```

or

```
$LKROOT/bin/sh $LKROOT/bin/getlocks.ksh args...
```

echo.exe does not translate character sequences

By default, the echo.exe utility in LifeKeeper v5.0 will not translate character sequences -- such as “\n” to Newline, “\t” to Tab. The echo.exe shipped with previous versions of LifeKeeper did perform the translation.

If your script depends on character sequence translations, and the script explicitly calls \$LKROOT/bin/echo, you have one of two options. Either use the ksh built-in echo utility, by removing the path specifier before echo (i.e. change “\$LKROOT/bin/echo” to “echo” so the shell will use its built-in function), or specify the “-e” option when calling \$LKROOT/bin/echo.

Filename case sensitivity with wildcard characters

In LifeKeeper v5.0, ksh.exe and sh.exe support case-insensitive access to files, if the file name is explicitly given. If the filename contains wildcard characters, however, the case of any characters which are provided must match the filename explicitly.

For example – if the following files exist in /tmp

```
a.txt b.txt C.TXT
```

then executing “ls /tmp/*.txt” will return “a.txt b.txt”. Executing “ls /tmp/*.TXT” will return “C.TXT”. Executing “ls /tmp/c.*” will not find any files. Executing “ls /tmp/c.txt” will return “c.txt”.

If your script searches for files using wildcards, and you need case insensitivity, then follow the following example, which will find all .TXT files in /tmp regardless of case:

```
ls /tmp/*.[tT][xX][tT]
```

Negative exit value from executable

The LifeKeeper v5.0 ksh will interpret a negative return value from an executable as “0”. Previous versions of LifeKeeper came with a ksh which allowed negative return values.

Negative exit value from shell script

The LifeKeeper v5.0 ksh will not allow a script to exit with a negative value. Previous versions of LifeKeeper came with a ksh which allowed negative exit values.

No automatic .ksh or .sh extension on argument to “sh” command

In previous versions of LifeKeeper, it was possible to execute a script by executing sh.exe with the script name – minus the .ksh or .sh extension – as the first argument. For instance:

```
$LKROOT/bin/sh $LKROOT/bin/getlocks
```

was a valid way to execute the getlocks.ksh script.

LifeKeeper v5.0’s ksh will not append the file extensions to its arguments. In order to execute getlocks in LifeKeeper v5.0, you should change your script to simply run it without calling “sh.exe”, or you should provide the full path and filename. Either of the following commands will work in LifeKeeper v5.0:

```
$LKROOT/bin/getlocks
```

or

```
$LKROOT/bin/sh $LKROOT/bin/getlocks.ksh
```