Windows Shell Action Command Library

A Guide to the BigFix® Action Shell Commands

BigFix, Inc.
Emeryville, CA

Last Modified: May 27, 2003

Compatible with
BigFix Enterprise Suite (BES) version 3.0
and
BigFix Consumer Client version 1.7

All inquiries regarding the foregoing should be addressed to:

BigFix, Inc.
5915 Hollis Street
Emeryville, CA 94608-2017

Contents

Preface

Conventions Used in this manual

This document makes use of the following conventions and nomenclature:

Convention	Use
`Mono-space`	A mono-spaced font is used to indicate examples of actions and expressions in the Relevance Language.
bold Italics	Bold Italics are used for the titles of manuals and other cited literature.
[brackets]	Brackets are used to indicate *optional* items in an expression. For instance, [of <parameter>] means that an **of** statement may be included in the expression at your option.
{braces}	Braces indicate a substitution syntax. Items in braces are evaluated and the result is substituted in the action expression. The braces are literal, that is, they should be typed in the expression.
<angle bracket>	Angle brackets are used to indicate action parameters. These can be static or formed from substituted relevance expressions.

Examples

Square bullets and a mono-spaced font denote examples of Actions as used in a Fixlet message. If you have a color version of this file, these square bullets are also red:

- `delete "c:\updates\q312456.exe"`

Introduction

This manual details the properties and operations of the BigFix Action Shell Commands. After a Fixlet message locates a potential problem on a computer, it may offer to fix the problem with an *Action Command*. This allows the user to quickly cure the problem, often with a single mouse-click.

This manual lists all the BigFix action commands, with examples of each.

Many action commands allow or require parameters. Those parameters can either be hard-coded (static) values or expressions that are evaluated by the BigFix relevance language. These "substitution variables" endow actions with great power and flexibility.

The BigFix Enterprise Suite and the BigFix Support Suite contain many of the same shell commands, but there are some differences. Except when otherwise noted, one can assume that commands listed here pertain to both products.

This guide only applies to Windows operating systems.

Using Action Scripts

Creating Action Scripts

A Fixlet action can be one of a variety of types. For example, a Fixlet action may be a Macintosh AppleScript, a Visual BASIC script, a JavaScript, or a BigFix Action Script. A BigFix Action Script has the advantage of relevance expressions, which provide a way to create actions that are precisely tuned to a given computer.

An Action Script is simply a set of actions that a Fixlet author can include in a Fixlet message to cure a given problem. Since a problem is typically discovered by examining a client machine with BigFix Inspectors, it is only natural that these same inspectors would be used to drive the solution. Inspectors are included in Action Scripts using substitution, as explained in the next section. Using relevance expressions in your actions ensures that the solution will address the exact problem that was detected. As a result, the action will automatically cause the Fixlet to retire, no longer relevant.

1. To create a script, use the BDE program and select **New Button** from the Content menu.
2. From the Action Type pull-down, select **Windows Fixlet Shell Script**.
3. Type in a prompt for the action.
4. In the action text box, type your action commands.

The Fixlet will now include a text prompt and a button or a hyperlink that will execute the action script. When the Fixlet becomes relevant, the user can activate your script by clicking on the action button.

Using Substitution

Substitution allows the Fixlet author to include relevance expressions in an Action. That is accomplished by placing the relevance expression in curly braces:

- `run "{pathname of regapp "excel.exe"}"`

  This example runs a program without knowing where it is located. A relevance expression evaluates the pathname automatically using the 'regapp' inspector.

- `pause while {exists running application "updater.exe"}`

  This action pauses until a program finishes executing, using the 'running application' inspector.

Substitution is not recursive – the BigFix application is expecting to find a single expression inside the curly braces. If it sees another left brace before it encounters a closing right brace, it treats it as an ordinary character:

- `echo {"a left brace: {"}`

  would send this string to output:

  a left brace: {

Therefore no special escape characters are necessary to represent a left brace. To output a literal right brace without ending the substitution, use a double character:

- `echo {"{a string inside braces}}"}`

  would send this string to output:

  {a string inside braces}

File System Commands

delete

Deletes the named file. Any action script with the delete command will terminate if the file exists but cannot be deleted. This can happen due to write protection or an attempt to delete from a CD-ROM, for instance. If the file does not exist at all, however, the action script will continue to execute.

SYNTAX

```
delete "<FileName>"
```

EXAMPLES

- `delete "c:\program files\bigsoftware\module.dll"`
- `delete "{name of drive of windows folder}\win.com"`

NOTE

It's good practice to enclose filenames in quotes to preserve spaces in the filenames. Without quotes, the file system will not be able to access those files with spaces in the path or file name.

copy

Copies the source file to the named destination file. An action script with the copy command terminates if the destination already exists or if the copy fails for any other reason such as when the destination file is busy.

SYNTAX

```
copy "<Source_FileName>" "<Destination_FileName>"
```

EXAMPLES

- `copy "{name of drive of windows folder}\win.com" "{name of drive of windows folder}\bigsoftware\win.com"`
- `delete "c:\windows\system\windir.dll" copy "__download\windir.dll" "c:\windows\system\windir.dll"`

This pair of Action commands deletes the target file (if it exists) before it performs the copy action.

move

Moves the source file to the named destination file. This command also gives the action author the ability to **rename** a file. An action script with the move command terminates if the destination already exists, if the source file doesn't exist, or if the move fails for any other reason.

SYNTAX

```
move "<Source_FileName>" "<Destination_FileName>"
```

EXAMPLES

- `move "c:\program files\bigsoftware\module.dll" "c:\temp\mod.dll"`
- `delete "c:\updates\q312456.exe" move "__download\q312456.exe" "c:\updates\q312456.exe"`

open

Opens the indicated file. Open uses the file association, as defined in the registry, to select the proper application to open the document.

SYNTAX

```
open "<Source_FileName>"

open <URL>
```

EXAMPLES

- `open http://www.MegaSoft.com/MegaSoft.fxm`

  This command will subscribe to the MegaSoft Fixlet site.

- `open "c:\program files\bigfix\bfast.exe"`

  Open the BFast application.

- `open http://yahoo.com`

  Open the default browser and go to the Yahoo site.

NOTE

Open has the same effect as issuing a ShellExecute("open", fileName)) statement from Windows. The argument to the open command is usually a filename, but it can also be generalized to a URL as well.

The open command can fail for any of the following reasons:

- The specified file was not found.
- The specified path was not found.
- The .exe file is invalid (non-Win32 .exe or error in .exe image).
- The operating system denied access to the specified file.
- The file name association is incomplete or invalid.
- The DDE transaction could not be completed because other DDE transactions were being processed.
- The DDE transaction failed.
- The DDE transaction could not be completed because the request timed out.
- The specified dynamic-link library was not found.
- There is no application associated with the given file name extension.
- There was not enough memory to complete the operation.
- A sharing violation occurred.

download

Downloads the file indicated by the URL. After downloading, the file is saved in a directory named "__download" (the name begins with two underscores) relative to the local folder of the Fixlet Site that issued the download command (by default, these local folders exist at "c:\program files\bigfix__Data\".

If the download fails, the action script terminates. The name of the file is derived from the part of the URL after the last slash.

For instance, consider the command:

- `download ftp://ftp.microsoft.com/deskapps/readme.txt`

The action example above downloads the readme.txt file from the Microsoft site and automatically saves it in the local __download folder as readme.txt.

SYNTAX

```
download < [options] File_URL ["progress_display.html"
[width height]]>
```

[options]

The download Command can be prefaced with two optional keywords, *open* and *save*. These may be used to give the end-user the ability to choose the download folder and/or open the file when the download completes. These options are ignored by the BigFix Enterprise System (BES).

[progress_display.html [width height]]

The download command URL can be followed by the name of an html file (usually part of the Fixlet site) that will be displayed in the progress indicator during the download. The width and height values provide the dimensions of the message display window in pixels. The default window size is 400 by 60 pixels. This optional banner is ignored by BES.

EXAMPLES

- `download http://download.bigfix.com/update/bf1504.exe`

  Downloads the bf1504.exe file from the BigFix site, and directs the downloaded file to the default site "__download" folder.

- `Download open`
  `http://download.bigfix.com/update/bf1504.exe`

  Directs the downloaded file to the default site "__download" folder and provides the user the option to immediately **open** the file once the download completes.

- `Download save`
  `http://download.bigfix.com/update/bf1504.exe`

  Prompts the user for the location to **save** the file before the download begins.

- `Download open save`
  `http://download.bigfix.com/update_patch.exe`

  Prompts the user for the **save** file location *and* adds an **open** button to the progress dialog when the download completes.

- `download`
  `"http://download.microsoft.com/download/prog.exe" run`
  `"__download\prog.exe"`

  This set of actions automates the download process. If it's not necessary to involve the user in the save and open process, you can use a **download** and **run** command to handle

the entire process for the user, reducing the application of an executable patch to a single click. Note that the downloaded program is run from the '__download' directory of the Fixlet site, where the download command places it. The Fixlet site directory is the working directory for all commands, and the __download directory is located there.

- ```
  Download
  "http://download.shareware.com/download/prog.exe"
  "file://{location of parent folder of masthead of
  current site}/ad.html" 468 65
  ```

This example puts up an ad that is displayed in a dialog box during the download period. The dialog box includes a progress bar to indicate the state of the download.

appendfile

The **appendfile** command creates a text file named **__appendfile** in the site directory, by default "C:\Program Files\BigFix__Data\<site name>. Each time you invoke the command, it appends the text you specify after the command to the end of the file. This command may be useful for creating diagnostic files or dynamically building files that incorporate attributes of the end-user's machine. The __appendfile is automatically deleted when the shell commands begin.

SYNTAX

```
appendfile <text [{substitutions}]>
```

EXAMPLES

- ```
  appendfile This file will contain details about your
  computer appendfile Operating System={name of operating
  system} appendfile Windows is installed on the {location
  of windows folder} drive
  ```

These commands record the OS and Windows location in the append file

- ```
  appendfile {("Disk " & name of it & ", free space=" &
  free space of it as string) of drives}
  ```

The above example records the name and the free space available for all the drives on the client PC.

NOTE

Although you could use a series of DOS commands to build a text file, the end-user would notice the side effect of the command window flashing each time a command was executed. By using the appendfile command to build the file and then the appropriate Action
  ```

commands to execute the program to process the file, you can create a clean solution that will be imperceptible to the user.

Instead of coding:

- `dos echo [HKR] > %windir%\smcfg.ini dos echo HostBasedModemData\Parameters\Driver,ModemOnHoldEnabled, 1,00,00>> %windir%\smcfg.ini dos smcfg`

You can use the following syntax to create the .ini file using BigFix commands rather thandos commands, avoiding the "flashing black command windows."

- `appendfile [HKR] appendfile HostBasedModemData\Parameters\Driver,ModemOn,1,00,00 delete {location of system folder}\smcfg.ini copy __appendfile {location of system folder}\smcfg.ini run smcfg`

browse to

Causes the web browser to navigate to the named URL. This can be used to bring up information relating to a proposed set of actions. In BES, this is treated the same as the open command.

SYNTAX

```
browse to <URL>
```

EXAMPLE

- `browse to http://support.microsoft.com/support/kb/articles/q253/9/12.asp`

Setting Commands

setting

Settings are named values that can be applied to individual Fixlet sites or to client computers. Each setting has a time associated with it. In the BES Console, settings can be created and propagated by the BigFix administrator. Settings can also be created by actions, as follows:

SYNTAX

```
setting "<name"="value" on "date" for client>

setting "<name"="value" on "date" for site "sitename>"
```

EXAMPLE

- `setting "name"="Bob" on “{now}” for client`

  Sets the namevariable to Bob on the client machine.

- `setting "preference"="red" on “{now}” for site "color_site"`

  Sets the preference variable to red for the specified site. Note that unless there are multiple sites with the same name, you can specify the site without the full gather url.

- `setting "time"="{now}" on “{now}” for current site`

  Sets the time variable to the current time on the current site.

- `setting “division”=”%22design group%22” on “{now}” for client`

  This example uses %xx to indicate special characters by their hexadecimal equivalent. In this case, %22 will enclose the value of the variable in double quotes.

‡ BigFix Enterprise Console Only.

setting delete

This action deletes a named setting variable on the client computer.

SYNTAX

```
setting delete ”<name>” on “<date>” for client
setting ”<name>” on “<date>” for site “<site_url>”
```

EXAMPLE

- `setting delete "name" on "{now}" for client`
- `setting delete "abc" on "{now}" for site "siteurl"`
- `setting delete "abc" on "{now}" for current site`

‡ BigFix Enterprise Console Only.

Registry Commands

regset

Sets a registry key to the given name and value. If the key doesn't already exist, this command creates the key with this starting value.

SYNTAX

```
regset "<registry key>" "<value name>"=<value>
```

These values are entered just as they are in a registry file, in keeping with the rules for Regedit, the Windows program that edits the registry. String entries are offset by quotes, and the standard 4-byte integer (dword) is entered in hexadecimal with leading zeroes.

EXAMPLES

- ```
  regset
  "[HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word\S
  ecurity]" "Level"=dword:00000002
  ```
- ```
  regset "[HKEY_CURRENT_USER\Software\BigFix Inc.]"
  "testString"="bob"
  ```
- ```
  regset "[HKEY_CLASSES_ROOT\ShellScrap]"
  "AlwaysShowExt"=""
  ```

  This example clears the data of a registry value.

NOTE

Notice in these examples that square brackets [ ] are used to enclose the name of the registry key. Again, this is in keeping with the rules for Regedit files. This syntax is necessary for the RegSet command, but not for registry Inspectors.

When you use the BigFix regset command, keep in mind that the BigFix client dynamically builds the .reg file that you would have had to create manually to update the registry and then it executes that resulting .reg file for you. One of the rules of the .reg file is that any \'s in the **value** field need to appear as double slashes, that is \\. So if you were trying to assign the value SourcePath2 of the registry key HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion to c:\I386, the command that you would define would look like this:
  ```

- `regset “[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion]” “SourcePath2”=”c:\\I386”`
- `regset “[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion]” “SourcePath2”={escape of ”c:\I386”}`

This example uses the **escape** relevance clause to automatically convert backslashes to double backslashes.

regdelete

Deletes a registry key value of the given name. If the value doesn’t already exist, this command will fail and all subsequent commands will not be executed.

SYNTAX

```
regdelete “<registry key>” “<value name>”
```

EXAMPLE

- `regdelete “[HKEY_CLASSES_ROOT\ShellScrap]” “NeverShowExt”`

NOTE

In order to delete a non-empty registry key and all its sub-keys, you need to create a file, say del.reg, that looks like this:

```
REGEDIT4

[-HKEY_CURRENT_USER\keep\removethisandbelow]
```

There should be three lines in this file, and the last line must be a blank. Note the dash (-) in front of the registry path. Now you can execute an action like this:

- `regedit /s del.reg`

When this action is executed, the key named removethisandbelow, along with all its sub-keys, is deleted.

Fixlet Maintenance Commands

fixlet delete

Deletes the Fixlet message that executes this shell command. The Fixlet message is moved to the trash and will no longer be evaluated for relevance. This Command is useful for Fixlet messages that have no relevance statements defined to make the message disappear automatically once the corrective action has been taken. All subsequent commands in the Action are ignored, so this must be the final command.

SYNTAX

```
fixlet delete
```

EXAMPLE

- `fixlet delete`

enable gathering

Turns off the 'Block Automatic Gathering' setting in the preferences dialog and initiates a gather for all subscribed Fixlet sites.

SYNTAX

```
enable gathering
```

EXAMPLE

- `enable gathering`

fixlet close

Closes the window of the Fixlet message performing the current Action. This action is ignored by BES.

SYNTAX

```
fixlet close
```

EXAMPLE

- `fixlet close`

fixlet restore

This command is used to automatically pull a Fixlet message out of the trash. The Fixlet message in the current site that matches the specified Fixlet ID is pulled from the trash. This is true even if it the trash has been emptied. If the Fixlet message does not exist, the command fails.

SYNTAX

```
fixlet restore <fixlet ID>
```

EXAMPLE

- `fixlet restore 104`

Execution Commands

dos

Issues a standard DOS command. If the DOS command fails, the action script that contains it is terminated.

SYNTAX

```
dos <DOS command line syntax>
```

EXAMPLE

- `dos rmdir /Q /S "{pathname of windows folder & "\temp\BigFixQ"}"`

  This example deletes a directory from a temporary folder in the windows directory.

- `dos scandisk.exe e:`

  In this example, e: is a parameter passed to the scandisk program.

NOTE

This has the same effect as issuing a system("DosCommandLine") statement from the Windows API. It is also the same as typing the DosCommandLine to a DOS prompt.

The DOS command uses the PATH environment variable to try to locate the command on the user's hard drive. If you want any it to look elsewhere, you must specify a complete pathname.

run

Executes the indicated program. If the process can't be created, the action script is terminated. Run does not wait for the process to terminate before executing the next line of the action script. The command line contains the name of the executable and may optionally contain parameters. If you want to wait for one program to finish before starting another one, use the wait command (see below).

SYNTAX

```
run <command line>
```

EXAMPLE

- `run "{pathname of regapp "excel.exe"}"`
- `run "c:\winnt\ftp.exe" ftp.sonic.net`
- `run wscript /e:vbs x.vbs arg1 arg2`

This example shows how you might run a script and pass it some arguments.

NOTE

This command has the same effect as issuing a CreateProcess("CommandLine") statement from the Windows API. This is also the same as using CommandLine in the Windows RUN dialog.

See the Windows documentation on CreateProcess for a discussion of the method used to locate the executable from a CommandLine.

rundetached

Rundetached is used to prevent pop-up DOS windows when you execute a program. It's the same as the **run** command, but the process created doesn't access the parent's console, which inhibits the annoying DOS window.

SYNTAX

```
rundetached <command line>
```

EXAMPLE

- `rundetached "{pathname of regapp "excel.exe"}"`
- `rundetached "c:\winnt\ftp.exe" ftp.sonic.net`
- `rundetached wscript /e:vbs x.vbs arg1 arg2`

This example shows how you might run a script and pass it some arguments.

NOTE

This command has the same effect as issuing a CreateProcess("CommandLine") statement from the Windows API. This is also the same as using CommandLine in the Windows RUN dialog. See the Windows documentation on CreateProcess for a discussion of the method used to locate the executable from a CommandLine.

wait

The wait command behaves the same as the run command, except that it waits for the completion of one process or program before continuing.

SYNTAX

```
wait <command line to execute program>
```

EXAMPLE

- `wait scandskw.exe`

NOTE

This has the same effect as issuing a CreateProcess("CommandLine") statement from the Windows API, and then waiting for completion.

waitdetached

Waitdetached is used to prevent pop-up DOS windows when waiting for a program to complete. It's the same as the **wait** command, but the process created doesn't access the parent's console, which inhibits the annoying DOS window.

SYNTAX

```
waitdetached <command line>
```

EXAMPLE

- `waitdetached scandskw.exe`
- `waitdetached wscript /e:vbs x.vbs arg1 arg2`

  This example shows how you might run a script, pass it some arguments and then wait for its completion.

NOTE

This has the same effect as issuing a CreateProcess("CommandLine") statement from the Windows API, and then waiting for completion.

script

Not to be confused with an action script, the **script** keyword executes an external script (created for a scripting language like JavaScript or Visual Basic) with the given name. The action script containing the **script** keyword will terminate if the appropriate scripting engine is not installed or if the script cannot be executed. The next line of the Action commands is not executed until the specified script terminates.

SYNTAX

```
script <script name>
```

EXAMPLE

- `script attrib.vbs`

NOTE

This has the same effect as issuing a wscript scriptName statement from Windows, and then waiting for completion. This is also the same as using scriptName from the Windows RUN dialog. If you need to pass parameters to your script, use the **run** command instead.

BigFix Client Maintenance Commands

module add

Adds the specified inspector library file to the set of inspector libraries to be used by the client. When replacing an inspector library, you must specify it in a module delete command as well as specifying it in the module add command.

SYNTAX

```
Module add "<module name>"
```

EXAMPLE

- `module add "dellinspect.dll"`

module delete

Marks the specified inspector library file for deletion.

SYNTAX

```
Module delete "<module name>"
```

EXAMPLE

- `Module delete "inspectors.dll"`

module commit

The add and delete commands set the stage for committing changes to the inspector libraries. The commit command actually performs the deletion and adding.

SYNTAX

```
Module commit
```

EXAMPLE

- `Delete "dellinspect.dll" copy "{pathname of client folder of site "dell"}\dellinspect.dll" "dellinspect.dll" module add "dellinspect.dll" module commit`

Comments

double forwardslash

Lines beginning with // are comments and are ignored during action execution.

SYNTAX

```
//
```

EXAMPLE

```
// The following command will replace the file on the C
drive copy "{name of drive of windows folder}\win.com"
"{name of drive of windows folder}\bigsoftware\win.com"
```

Flow Control Commands

continue if

The command will continue to the next command in the Command if the value provided as a parameter evaluates to true. It will stop without error if the specified value evaluates to false. You can use relevance substitution to compute the value.

SYNTAX

```
continue if <{relevance condition to evaluate}>
```

EXAMPLE

```
continue if {name of operating system = “Win2k”}
download http://www.real-
time.com/downloads/win98/dun40.exe
continue if {(size of it = 325904 and sha1 of it =
“013e48a5e71acb10563068fbdd69955b893569dc”) of file
“dun40.exe” of folder “__download”}
wait __download/dun40.exe /Q:A /R:N
action requires restart
```

pause while

The action will not continue to the next command while the relevance expression specified evaluates to true. It will continue and execute the next command of the Action as soon as the value evaluates to false or the value fails to evaluate. Use relevance substitution syntax to define the condition.

SYNTAX

```
pause while <{relevance condition to evaluate}>
```

EXAMPLE

```
pause while {exists running application “updater.exe”}
```

```
wait “C:\70sp3\msolap\install\setup.exe” -s -a -s -
f1”C:\70sp3\msolap\install\sql7olapsp3.iss” -
f2”C:\70sp3\result.log”
pause while {not exists file “C:\70sp3\result.log”}
pause while {not exists section “ResponseResult” of file
“C:\70sp3\result.log”}
```

action requires restart

The '**action requires restart**' command informs the client that the current action will not be completed until the next restart completes. Once this action has been completed on a machine, the inspector 'pending restart' will return 'True'. If there is an 'action requires restart' command in an action, the BigFix Enterprise Console will report 'Pending Restart' until the affected machine is restarted. This command is ignored by BSS.

SYNTAX

```
action requires restart
```

EXAMPLE

- `action requires restart`

action may require restart

When the BES-only command '**action may require restart'** is executed, the client looks at the system for telltale signs that a restart is needed. If so, it sets the action completion status such that the action will appear as 'Pending Restart' in the console, until a restart occurs. Once the restart is completed, the action completion status of the action will take on the value of 'success' if the relevance of the action is no longer relevant, or 'failed' if it is still relevant.

If the telltale signs of restart are not present, the action completion status of the action will take on the value of 'success' if the relevance of the action is no longer relevant, or 'failed' if it is still relevant.

SYNTAX

```
action may require restart
```

EXAMPLE

- `action may require restart`

‡ BigFix Enterprise Console Only.

action requires login

'**Action requires login**' informs the client that the current action will not be completed until the computer is restarted and an administrator logs in. Once this action has been completed on a machine, the inspector 'pending login' will return 'true'. When running the BigFix Enterprise Suite, if there is an 'action requires login' command in an action, the BigFix

Enterprise Console will return 'pending Login' until the computer is restarted and a user with administrative privileges logs in. This command is ignored by BSS.

SYNTAX

```
action requires login
```

EXAMPLE

- `action requires login`

action parameter query

This allows data entry of parameters to be available via relevance during action execution. Parameter names may include blanks, and are case sensitive. The parameter name, description, and value must each be enclosed inside double quotation marks (“). Once entered, the user input becomes the default in subsequent invocations.

SYNTAX

```
action parameter query "<parameter name>" [with description
"<description>"] [and] [with default [value] "<default
value>"]
```

EXAMPLE

- `action parameter query “InstallationPoint” with description “Please enter the location of the shared installation point:”`
- `action parameter query “Registry key” with description “Please enter your desired registry key” and with default value “null”`
- `action parameter query “tips” with description “Enter ‘on’ or ‘off’ to control Fixlet tips.” With default “on” regset “[HKEY_CURRENT_USER\Software\BigFix]” “tips”=”{parameter “tips” of action}”`

NOTE

The parameter values input by the user may include %xx where xx stands for a two-digit hexadecimal number to specify the character you want to embed. To embed a percent sign, use %25. To embed a double quote, use %22.

To retrieve the action parameter value, for example in relevance substitution use: {parameter “parameter name” of action}

set clock

Causes the client to re-register with the registration server, and to sets its clock to the time received from the server during the interaction. This is useful when the client's clock is out of sync. This BES-only command is not available when the client is operating under an evaluation license.

SYNTAX

```
set clock
```

EXAMPLE

- `set clock`

‡ BigFix Enterprise Console Only.

restart

The **restart** command will restart the computer. If the optional <delay seconds> parameter is provided, the shutdown will happen automatically after the specified delay.

If a user is logged in, a UI will be displayed that shows the delay counting down. In this case, the UI will have a **Restart Now** button instead of a **Cancel** button.

If the delay parameter is not specified, the user is prompted to press a button to restart the computer.

SYNTAX

```
restart [<delay seconds>]
```

EXAMPLE

- `restart 180`

NOTE

The delayed restart is a forced restart; it will not prompt the user to save changes to documents, etc. The machine will restart without further prompting.

‡ BigFix Enterprise Console Only.

shutdown

The **shutdown** command is similar to the **restart** command, but it simply shuts the computer down and does not reboot.

If the optional <delay seconds> parameter is provided, the shutdown will happen automatically after the specified delay.

If a user is logged in, a UI will be displayed that shows the delay counting down. In this case, the UI will have a **Shutdown Now** button instead of a **Cancel** button.

If the delay parameter is not specified, the user is prompted to press a button to shut down the computer.

SYNTAX

```
shutdown [<delay seconds>]
```

EXAMPLE

- `shutdown 180`

NOTE

The delayed shutdown is a forced restart; it will not prompt the user to save changes to documents, etc. The machine will shut down without further prompting.

‡ BigFix Enterprise Console Only.

Administrative Rights Commands

administrative rights enable

The **administrative rights enable** command lets you 'turn on' administrative rights for the targeted computer(s). Once enabled, you can then grant administrative rights to specific personnel. This is accomplished by using a setting with an an effective date, passed as a parameter. The date is not optional. The effective date tests are the same as for ordinary settings.

SYNTAX

```
administrative rights enable on [date]
```

EXAMPLE

- `administrative rights enable on "21 Aug 2002 17:39:14 gmt"`

  Sets the __AdministrativeRights setting to "enabled".

‡ BigFix Enterprise Console Only.

administrative rights disable

The **administrative rights disable** command lets you 'turn off' administrative rights for the targeted computer(s). Once disabled, you can no longer grant administrative rights to specific personnel. Disabling is accomplished by using a setting with an an effective date, passed as a parameter. The date is not optional. The effective date tests are the same as for ordinary settings.

SYNTAX

```
administrative rights disable on [date]
```

EXAMPLE

- `administrative rights disable on "21 Aug 2002 17:39:14 gmt"`

  Sets the __AdministrativeRights setting to "disabled".

‡ BigFix Enterprise Console Only.

administrator add

The **administrator add** command lets you appoint specific people to administer specific BES Clients. This is accomplished by using a setting with an an effective date, passed as a parameter. The date is not optional. The effective date tests are the same as for ordinary settings.

SYNTAX

```
administrator add [administator name] on [date]
```

EXAMPLE

- `administrator add "bob" on "21 Aug 2002 17:39:14 gmt"`

  Allows the BES Console operator named 'bob' to have adminstrative rights on the targeted computer, effective on the given date.

‡ BigFix Enterprise Console Only.

administrator delete

The **administrator delete** command lets you revoke administrative rights for the specified administrator. This is accomplished by using a setting with an an effective date, passed as a parameter. The date is not optional. The effective date tests are the same as for ordinary settings.

SYNTAX

```
administrator delete [administator name] on [date]
```

EXAMPLE

- `administrator delete "bob" on "21 Aug 2002 17:39:14 gmt"`

  Revokes the administrative rights of the BES Console operator named 'bob', effective on the given date.

‡ BigFix Enterprise Console Only.

Locking Commands

action lock until

Locks actions from the **effective date** until the expiration date occurs. The expiration date is MIME time format (as in 19 Jul 2002 12:42:51 -0700).

SYNTAX

```
action lock until "<expire date>" "<effective date>"
```

EXAMPLE

- `action lock until "{now + 3*days}" "{now}"`

‡ BigFix Enterprise Console Only.

action lock indefinite

Turns on action lock, starting on **effective date**, which will never expire. The date is in MIME time format (as in 19 Jul 2002 12:42:51 -0700).

SYNTAX

```
action lock indefinite "<effective date>"
```

EXAMPLE

- `action lock indefinite "{now}"`

‡ BigFix Enterprise Console Only.

action unlock

Unlocks the client to act upon any actions. The effective date field is used to insure that locking and unlocking actions take place in the order in which they were created. The date is in MIME time format (as in 19 Jul 2002 12:42:51 -0700).

SYNTAX

```
action unlock "<effective date>"
```

EXAMPLE

- `action unlock "{now}"`

‡ BigFix Enterprise Console Only.

Site Maintenance Commands

site force evaluation

Causes the client to re-evaluate all Fixlet messages for the site.

SYNTAX

```
site force evaluation
```

EXAMPLE

- `site force evaluation`

site gather schedule publisher

This BES-only command sets the schedule for gathering the origin site to that specified in the masthead for the site.

SYNTAX

```
site gather schedule publisher
```

EXAMPLE

- `site gather schedule publisher`

‡ BigFix Enterprise Console Only.

site gather schedule manual

This BES-only command disables scheduled gathering from the origin site.

SYNTAX

```
site gather schedule manual
```

EXAMPLE

- `site gather schedule manual`

‡ BigFix Enterprise Console Only.

site gather schedule disable

This BES-only command disables scheduled gathering from the origin site.

SYNTAX

```
site gather schedule disable
```

EXAMPLE

- `site gather schedule disable`

‡ BigFix Enterprise Console Only.

site gather schedule seconds

This BES-only command sets the schedule for gathering from the origin site to the number of seconds specified.

SYNTAX

```
site gather schedule seconds <seconds>
```

EXAMPLE

- `site gather schedule seconds 360`

‡ BigFix Enterprise Console Only.

subscribe

Subscribes the client to the site identified in the masthead file.

SYNTAX

```
subscribe "<masthead file name>"
```

EXAMPLE

- `subscribe "__download\Sitename.fxm"`

unsubscribe

Automatically unsubscribes from the current site. The user is prompted to confirm the unsubscribe Action. The action site itself can't be unsubscribed, and the action fails if it is attempted.

SYNTAX

```
unsubscribe
```

EXAMPLE

- `unsubscribe`

Index

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Y