BigFix Version 10.0.1
Action Script Guide

Special notice

Before using this information and the product it supports, read the information in Notices (on page 87).

Contents

Chapter 1. The Action Script Language

The scope of the Action Script language is to issue commands, named **actions**, from within Fixlets and tasks on relevant clients to fix the problem identified by the Applicability Relevance clause.

In a Fixlet or task message you can specify an action script including one or more actions. These actions will be run in sequence on relevant clients when the console operator clicks **Take action** in the Fixlet or in the task entry on the BigFix console. If an action fails running, the subsequent actions will not run and the processing will stop. For this reason, it is very important to use error prevention methods in action scripts.

One of the biggest strengths of the Action Script language is the ability to use relevance language expressions to implement variables within actions. This ability, named substitution, allows you to customize an action for each specific agent where the action runs, for example, by resolving the local installation path of a proprietary application.

The information about the Action Script Language is divided into the following areas:

- Guide: Where you find the generic concepts that apply to the action script language.
- Reference: Which contains information about all the available statements.

Chapter 2. Guide

Here you find the main concepts to understand how to use the Action Script Language.

Creating a Custom Action

You can create custom actions to fix specific problems or address issues across your network that are not covered by the standard content.

To create a custom action:

1. Log on to the BigFix Console as a Master Operator.
2. Select **Tools > Take Custom action**.
3. In the **Take action** dialog provide a **Name** for your custom action. The value in this field can be sorted and filtered, keep it in mind when defining your naming convention.
4. The Preset pull-down menu allows you to choose a preset customized action. These are the the operations that you can run against a preset action:
    - **Preset:** Select a preset from the pull-down menu.
    - **Show only personal presets:** Check this box to filter the list of presets to just your personal ones.
    - **Save Preset:** Save the current set of action options for later use. A check box below that lets you save it as a public or private preset.
    - **Delete Preset:** Removes this preset from the selectable list.
5. Clicking on the different tabs you can define and customize the preset action:
    - **Target:** Select the targets from the provided list, or use properties or a specific list of computers to target the action.
    - **Execution:** Specify the deployment options and constraints, including repeated application and failure recovery.
    - **Users:** Determine how this action will respond to the presence or absence of users.
    - **Messages:** Provide a message to precede and accompany the action.
    - **Offer:** Create an action offering, allowing the user to choose whether or not to apply the action.

- **Post-action:** Describe what actions need to be done to complete the action, including restarts or shutdowns.
- **Applicability:** Allows you to override the original action relevance.
- **Success Criteria:** Create specific criteria that you can use to determine if your action was successful.
- **Action Script:** This tab allows you to create or modify an action script.

6. When you finish customizing the custom action and you are ready to deploy it, click **OK**.

Your custom action will be distributed to all the computers that have been selected or targeted. The actions will be applied using whatever constraints and schedules that you have specified.

You can also create actions when you Create Tasks or Create Fixlets.

Note: The original action included in the Fixlet or in the task is not overwritten by your custom action.

Using Substitution Variables

Substitution allows the Fixlet author to include relevance expressions in an action. This is accomplished by placing the relevance expression in curly braces.

For example, this example runs a program without knowing where it is located. A relevance expression evaluates the path name automatically using the ‘regapp’ inspector:

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

In this example, instead, the action pauses until a program finishes running:

```
pause while {exists running application "c:\updater.exe"}
```

BigFix expects 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. For example, the output of the action:

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

would be:

```
a left brace: {
```

This means that 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, for example:

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

would return:

```
{a string inside braces}
```

This is another example:

```
appendfile {{ name of operating system } {name of operating system}
```

When this example is parsed, the double left braces indicate that what follows is not a relevance expression. Only a single right brace is necessary when it's outside of a relevance expression (inside a relevance expression, a double right brace is necessary to specify a literal one). This would output the following line to __appendfile:

```
{ name of operating system } WinXP
```

You can also use substitution with add prefetch item commands in prefetch blocks, for example:

```
begin prefetch block
   parameter "manifest"="{pathname of file "manifest.spec" of client folder
 of site "AV"}"
   add prefetch item {concatenation " ; " of lines of file (parameter
 "manifest")}
end prefetch block
```

The Prefetch Block Structure

The prefetch block must be the first entry in the action script. It contains all the download prefetch logic needed to prepare for subsequent action execution. The instructions contained in the prefetch block must successfully complete before the rest of the action can continue; in this way it is ensured that files are successfully downloaded before the action script runs.

The prefetch block structure must satisfy the following criteria:

- Is located as first entry in the action script. Only blank lines and comments are allowed to precede it.
- Starts with a **begin prefetch block** statement.
- Ends with the **end prefetch block** statement.

Note: Only one prefetch block is allowed per action.

Some of the methods that can be used in a prefetch block include:

Literal downloads : These are ordinary static downloads, which are still available.

Conditional downloads : Only those commands inside TRUE condition pathways are performed.

Variable Substitution : This includes downloads that use relevance substitution to determine which files to collect.

Custom logic : This takes advantage of a plug-in to create download manifests.

Static Download

Before it runs an action, the BigFix Client parses it, looking for download or prefetch commands.

Static downloads include the URL, the SHA hash algorithm, and the size for each item as literal values in the action script. The literal values allows an operator to see exactly what the action script is going to do. These literals are used to construct a numbered list of downloads associated with the action that is then stored on the BigFix Server. This stage of action processing is called **prefetch processing**.

To easily create prefetch commands, you can use the utility:

make-prefetch

As a consequence of prefetch processing, the Client notifies the nearest Relay of the need for downloads by requesting a URL ending in <actionid>/0, which in turn triggers the Relay to download all the items corresponding to that specified action. When they are ready, the Relay pings the clients back with the action ID. All the BigFix Clients running that action will then collect the files by asking for them one at a time as <actionid>/1, <actionid>/2, and so on.

However, because the download information is represented by literal expressions, only those URLs already known when the action is authored can be represented. This means that static downloads cannot be used for those instances where the downloads change, but the action script remains the same.

Dynamic Download

Dynamic downloads add the ability to use relevance clauses to specify downloads. These new commands must be embedded in a special segment of action code called a prefetch block. The prefetch block structure ensures that the file is successfully downloaded before the action script runs.

Note: Only one prefetch block is allowed per action.

The following examples show how to use the **prefetch block** to run dynamic downloads.

In this example, a file named `download.spec`, containing a named variable in its first line, is created in the AV Fixlet site:

```
name=update.exe sha1=123 sha256=678 size=456 url=http://site.com/download/
patch.exe
```

You can access the patch referenced in the download.spec file by using the relevance substitution in the prefetch block of the action script:

```
begin prefetch block
```

```
   // Creates a variable named downloadFile that points to a file in the AV
 site.
   parameter "downloadFile"="{pathname of file "download.spec" of client
 folder of site "AV"}"

   // Adds this file to the prefetch queue for subsequent downloading.
   add prefetch item {line 1 of file (parameter "downloadFile")}
end prefetch block
```

In this way, a Fixlet message in the AV site could offer to keep something automatically updated and the download.spec file would be refreshed whenever a new version becomes available.

Another popular technique is to use a data file, or *manifest*, containing a list of multiple downloads, each with its own URL, SHA hash algorithm, and size. This is useful when the files to download change often, as in updated spy ware or anti-virus definitions. This is an example of a manifest file:

```
name=patch1.exe sha1=123 sha256=347 size=456 url=http://site.com/download/
patch1.exe
name=patch2.exe sha1=234 sha256=358 size=567 url=http://site.com/download/
patch2.exe
name=patch3.exe sha1=345 sha256=368 size=678 url=http://site.com/download/
patch3.exe
```

You can download these patches with a prefetch block that pulls these files from the manifest, for example:

```
begin prefetch block
   parameter "manifest"="{pathname of file "manifest.spec" of client folder
 of site "AV"}"
   add prefetch item {concatenation " ; " of lines of file (parameter
 "manifest")}
end prefetch block
```

You can also use the **execute prefetch plug-in** command to use small executables to process files into a fresh manifest, for example:

```
begin prefetch block
   // Adds the plugin to the prefetch queue
   add prefetch item name=myPlugIn.exe sha1=123 size=456 url=http://mysite/
plugin.exe sha2=347

   // Collects the plug-in before prefetch processing continues
   collect prefetch items
   parameter "ini"="{file "prepass.ini" of site (value of setting
 "CustomSite") of client}"

   // Runs the plug-in with its arguments including the path for the data
   // file and the manifest to be produced from it.
   execute prefetch plug-in "{download path "myPlugIn.exe"}" /downloads
 "{parameter "ini"}"
   "{download path "manifest"}"

   // Queues up the downloads specified in the freshly created manifest

   add prefetch item {concatenation " ; " of lines of download file
 "manifest"}
end prefetch block
```

A technique like this might also be used to decrypt a secure file into a plain-text manifest.

Dynamic downloads must specify files with the confirmation of a size or SHA hash algorithm. The URL, size, and SHA hash algorithm can come from a source outside of the action script. For dynamic downloading, BigFix uses a white-list of URLs to ensure that only authorized URLs can download files. This is the path to the white list:

```
<BES Server Install Path>\Mirror Server\Config\DownloadWhitelist.txt.
```

This file contains a newline-separated list of regular expressions using a Perl regex format, such as:

```
http://.*\.sitename\.com/.*
http://software\.sitename\.com/.*
http://download\.sitename\.com/patches/JustThisOneFile\.qfx
```

The first line is the least restrictive, allowing any file at the sitename domain to be downloaded. The second line requires a specific domain host and the third line is the most restrictive, limiting the URL to a single file named "JustThisOneFile.qfx".

An empty or non-existent white-list causes all dynamic downloads to fail. A white-list entry of ".*" (dot star) allows any URL to be downloaded.

Prefetch blocks allow conditional statements:

```
begin prefetch block
   if {name of operating system = "Windows 2000"}
      add prefetch item name=up.exe sha1=123 size=456 url=http://site.com/
patch2k.exe sha2=567
   else
      add prefetch item name=up.exe sha1=123 size=456 url=http://site.com/
patch.exe sha2=567
   endif
end prefetch block
wait "{download path "up.exe"}"
```

This action script branches on the existence of Win2K, but the downloads in this example are described statically (as literal text). Although the clients will only download the particular items they need, all the static files are downloaded to servers and relays as soon as they are requested.

Dynamic downloads can improve this situation because only those files actually needed by clients are retrieved by to the server and relay in the first place. Here's an example using dynamic downloading:

```
begin prefetch block
```

```
   if {name of operating system = "Windows 2000"}
      add prefetch item {"name=up.exe sha1=123 size=456 url=http://
site.com/patch2k.exe"} sha2=567
   else
      add prefetch item {"name=up.exe sha1=123 size=456 url=http://
site.com/patch.exe"} sha2=567
   endif
end prefetch block
wait "{download path "up.exe"}"
```

By using relevance substitution in the prefetch block, with a properly configured white-list file on the server, this code only fetches the necessary file, potentially improving bandwidth requirements and efficiency.

You can also branch execution based on the contents of a file, allowing you to automate updates. This can be especially useful for dealing with changing version numbers. For example, you could create a file named 'manifest.txt' containing two named variables, such as:

```
version=1234
download=name=update.exe sha1=123 size=456
url=http://site.com/download/patch.exe sha2=567
```

Note that the download variable contains the name, sha1, sha2, size and URL of the patch file.

You can then use relevance substitution to extract these variables with an expression, such as:

```
parameter "ver"="{key "version" of file "{download path "manifest.txt"}"}"
parameter "filename"={key "download" of file "{download path
 "manifest.txt"}"}
```

By comparing the extracted version against some stored values, you can determine if and when you need to download the specified file. This technique can be expanded to

include multiple versions and can even be used to distinguish between patches and full replacement updates.

No matter which technique is used, after the files have been downloaded, they can be examined with various Inspectors. Before the action runs, these files are collected in a prefetch folder. While the action is running, they are located in the __Download folder.

These Inspectors can be used to locate the files before or while the action runs:

- **download folder:** During the prefetch parsing, this Inspector returns a folder object from the `__Global\<sitename>\<actionid>\named folder`.
- **download path "pathname":** This Inspector returns a string containing the full pathname to the specified file, whether it exists or not. The download filename is equivalent to `(pathname of download folder) & <pathseparator> & filename`.
- **download file "filename":** This Inspector returns a file object from the download folder or another named folder. The download filename is equivalent to `file 'filename' of download folder`.

The action script author must protect users from these actions and ensure that downloads and their checksums are not been compromised. An end-to-end authentication mechanism resistant to man-in-the-middle attacks is the best defense. When authoring a dynamic download action, it is critical to craft the action so that it authenticates information before using it, typically by using a plug-in as described above. It is also a good practice to explicitly identify those steps in the action script that perform this authentication so that users of your action can audit the mechanism before deciding to trust it.

Action Status Messages

Actions might report the following statuses back to the BigFix Server while processing on the client:

Not Reported

No report on this action yet. No report has yet been received from the endpoint for the action taken. We cannot confirm if the action has been

propagated, mirrored, gathered, processed, or reported until this status changes to something else.

Fixed

The action executed successfully. The BigFix Client has run the action and the relevance is now false (meaning that the action ran and fixed the issue).

Running

The action is currently running.

Evaluating

Evaluating relevance and action constraints. The BigFix Client has received the action targeted at it and will evaluate the action to see if it is time to run, the issue is still relevant, and so on.

Completed

The action has completed and no other actions are required.

Failed

The action failed. The BigFix Client has run the action and the issue is still relevant (even if the action ran successfully). Note that in the cases of patches, 'Failed' usually means the patch file was run, but failed to actually patch the computer.

Cancelled

The action was canceled by the user. The user clicked the "cancel" button when prompted with a message box.

Download Failed

A required download failed.

Locked

This computer is locked. The BigFix Client is in the "Locked" state that prevents it from running actions until unlocked.

Waiting

The BigFix Client is waiting for some condition to be able to run the action. The waiting conditions include: waiting for user input, waiting to retry after failure, waiting for a time/date range, waiting for a distribution time, waiting for a user to log in, and waiting until the custom constraints property becomes relevant.

- Action has failed and is waiting before trying again.
- Waiting on action dependency.
- Waiting to run in specified time range.
- Waiting until the action start time.
- This computer is not licensed.
- Waiting to satisfy temporal distribution time constraint.
- Waiting for active user condition.

Pending Downloads

Waiting for downloads to be mirrored. The BigFix Client is waiting to receive the complete file. This state will persist until the download is available on the BigFix Server -> BigFix Relay -> BigFix Client.

Pending Restart

Waiting for restart to complete action. The action was completed, but the action status of 'Fixed' or 'Failed' cannot be assessed until the computer is restarted.

Pending Message

Waiting for user to respond to message.

Pending Login

Waiting for user to log in.

Constrained

The computer doesn't meet the specified retrieved property constraint.

Postponed

The user postponed execution of this action.

Invalid Signature

The client was unable to verify the signature on this action.

Not Relevant

The Fixlet that this action addresses is not relevant on this machine. Before running the action, the BigFix Client checked the relevance for the action and it is no longer true.

Pending Offer Acceptance

Waiting for user to accept this offer.

Offers Disabled

No user is able to accept this offer.

Disk Limited

The download size exceeds the maximum value set in the client setting *_BESClient_Download_PreCacheStageDiskLimitMB*, which can be modified through the Edit Computer Settings dialog.

Disk Free Limited

The remaining disk space is smaller than the value set in the client setting *_BESClient_Download_MinimumDiskFreeMB*. For the download to complete, space must be cleared on the endpoint, or the client setting must be changed using the Edit Computer Settings dialog.

Hash Mismatch

The download completed, but the file failed a hash comparison. To troubleshoot, investigate the network between the agent and its parent to eliminate network problems.

Transcoding Error

The action failed transcoding from the deployment codepage.

Pending Client Restart

Waiting for client restart to complete action.

error

- An unknown error occurred.
- The Fixlet context is missing or invalid.
- Invalid site context. The Fixlet site might no longer exist.
- Invalid action content: the action is empty.
- Invalid action content: the action type is invalid.
- Invalid action content: the action script contains a syntax error.
- This action contained invalid download syntax.
- The download manager encountered a configuration error.
- This action was not executed for unknown reasons.
- This action was run, but could not be restarted due to a client UI translation error.
- This action was not executed due to an error encountered while translating the client UI elements.
- This action was not executed due to an error showing the client UI.
- This action failed to complete because the Management Extender plug-in reported an error.
- This action was not executed because the operator who created it is not an administrator of this client.

Note: After an action expires, the action status is no longer updated.

Chapter 3. Action Script Language Reference

Client Commands

These commands can be used to control the behavior of the BigFix client.

action lock indefinite

This command locks the client starting on the effective date.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
action lock indefinite <date>
```

Example

Lock the client now.

```
action lock indefinite {now}
```

action lock until

This command locks the client starting on the effective date, and unlocks the client when the expiration date occurs.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

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

Examples

Lock the client immediately, unlocking in three days.

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

Lock the client for 10 minutes using the current apparent registration server time, which is based on the last time the client registered with the server.

```
action lock until "{apparent registration server time + 10 * minutes}"
"{apparent registration server time}"
```

action log all

This command tells the client to log all commands along with their parameters. This is the default behavior. This can be used to undo a previous action log command.

Version	Platforms
8.2.474.0	All

Syntax

```
action log all
```

Examples

Don't log the parameters of the first setting command, then restore the default logging behavior.

```
action log command
setting "secret"="hodor" on "{now}" for client
action log all
setting "normal"="winterfell" on "{now}" for client
```

action log command

This command tells the client to only log the commands of the action. The parameters of the commands in the action will not be logged.

Ordinarily all aspects of an action are logged, including commands and parameters. The parameters may contain information about establishing private keys or decrypting passwords. This command can be used to avoid logging such sensitive information.

Version	Platforms
8.2.474.0	All

Syntax

```
action log command
```

Examples

Don't log the parameters of the setting command.

```
action log command
setting "name"="Bob" on "{now}" for client
```

action unlock

This command unlocks the client. The effective date field is used to ensure that locking and unlocking actions take place in the order in which they were created.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
action unlock <date>
```

Examples

Unlocks the client immediately.

```
action unlock "{now}"
```

administrator add

This command lets you add a BigFix user as an administrator of the computer. This is accomplished by using a BigFix client setting with an effective date passed as a parameter. The date is not optional. The effective date tests are the same as for the setting command.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
administrator add <operator-name> on <date>
```

Examples

Add the BigFix operator named `bob` as an administrator of the client computer.

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

Notes: The `operator-name` that this command expects is the masthead user name of the operator. To determine what the masthead user name is for an operator, you can use the masthead operator name session inspector.

administrator delete

This command allows you to remove a BigFix user as an administrator of the computer. 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 the setting command.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
administrator delete <operator-name> on <date>
```

Examples

Remove the BigFix operator named `bob` as an administrator of the client computer.

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

Notes: The operator-name that this command expects is the masthead user name of the operator. To determine what the masthead user name is for an operator, you can use the masthead operator name session inspector.

client restart

This command will restart the BES Client. It must be added as the last command in an action script or the command will fail.

Version	Platforms
9.0	Windows

Syntax

```
client restart
```

notify client

This command is equivalent to right clicking on a computer in the BigFix Console and selecting **Send Refresh**. This command may be necessary if the client is unable to receive notifications, which might happen if it can't receive UDP messages.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu.

Syntax

```
notify client ForceRefresh
```

plugin store

This command inserts, updates or deletes a BigFix cloud plugin setting. Plugin store settings are named values that can be encrypted or otherwise. Each plugin store setting has a timestamp associated with it.

Version	Platforms
10.0.0.0	Red Hat, Windows

Syntax

```
plugin store "<pluginName>" set "<pluginKey>" value "<value>" on "<date>"
plugin store "<pluginName>" set encrypted "<pluginKey>" value "<value>" on
 "<date>"
plugin store "<pluginName>" delete "<pluginKey>"
plugin store "<pluginName>" delete all
```

Where: `pluginName` describes the name of the plugin `pluginKey` describes the key name of the plugin `value` is the value to set `date` is a timestamp

and

`set` is the command to insert or update a plugin key value `set encrypted` is the command to insert or update a plugin key value in encrypted mode `delete` is the command to delete plugin keys `delete all` is the command to delete all plugin keys

Examples

```
plugin store "AWSPlugin" set "UName" value "JUser" on "31 Jan 2007 21:09:36
 gmt"
plugin store "AWSPlugin" set encrypted "UPassword" value "W34dfT_ghy7" on
 "{now}"
plugin store "AWSPlugin" delete "UPassword"
plugin store "AWSPlugin" delete all
```

relay select

This command issues a request to the client to perform a relay selection at the next opportunity. It always succeeds immediately, regardless of the success or failure of the pending relay selection.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
relay select
```

Examples

Request a relay selection.

```
relay select
```

restart

This 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 dialog will be displayed that shows the delay counting down. In this case, the interface will have a Restart Now button instead of a Cancel button.

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

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
restart [delay-seconds]
```

Where `delay-seconds` is an optional parameter to provide a delay before restarting.

Examples

Restart the computer in three minutes.

```
restart 180
```

Notes: The delayed restart is a forced restart. It will not prompt the user to save changes to documents. The machine will restart without further prompting.

set clock

This command causes the client to re-register with the server, and to sets its clock to the time received from the server during the interaction. This is useful when the client computer's clock is out of sync.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
set clock
```

Examples

Sychronize the client computer's clock with the BigFix server.

```
set clock
```

Notes: This command is not available when the client is operating under an evaluation license.

setting

This command sets a BigFix client setting.

Settings are named values that can be applied to individual sites or to client computers. Each setting has a timestamp associated with it. This timestamp is used to establish priority -- the latest setting will trump any earlier ones.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

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

Where `name=value` describes the setting, and `date` is a timestamp used to establish priority between conflicting `setting` commands.

Examples

Sets the setting `name` to `Bob` with an effective date of `31 Jan 2007 21:09:36 gmt`. It will supersede any other `name` setting with an earlier date.

```
setting "name"="Bob" on "31 Jan 2007 21:09:36 gmt" for client
```

Sets the preference setting to red for the site named color. There can be a different preference setting on each site. This example uses the now inspector to set the effective date to the time the action was evaluated.

```
setting "preference"="red" on "{now}" for site "color"
```

This sets the division setting to "design group". Note that the quotes are percent encoded.

```
setting "division"="%22design group%22" on "{now}" for current site
```

Notes:

When a client is reset, the effective dates of the settings are removed and any subsequent setting commands will overwrite them. There are several ways that clients can be reset, including computer-ID collisions (most often caused by accidentally including the computer ID in an image that gets copied to multiple systems), changing the masthead to a new server, or instructing the client to collect a new ID.

The actions that run next will establish a new effective date, but the setting values will be the same as before the reset. The values are retained because they contain information

such as relay selections. That way, when a deployment reset occurs, you don't have to issue new actions to reset your network relay structure.

setting delete

This command deletes a named setting variable on the client computer. It includes a timestamp which will be compared to the timestamp on the original setting. If the delete date is later than the setting date, the setting will be deleted. Otherwise, the delete command will be ignored.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
setting delete "<name>" on "<date>" for client
setting delete "<name>" on "<date>" for current site
setting delete "<name>" on "<date>" for site "<site_url>"
```

Examples

Deletes the `name` variable on the client machine.

```
setting delete "name" on "{now}" for client
```

shutdown

This 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

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
shutdown [delay-seconds]
```

Where `delay-seconds` is an optional parameter to provide a delay before shutting down.

Examples

This command will shut down the computer in three minutes.

```
shutdown 180
```

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

Download Commands

These commands allow you to download files to the BigFix client system.

add nohash prefetch item

This command downloads a file without verifying the file's hash. This is insecure and not recommended, but it might be necessary in some cases. It must be between a begin prefetch block and an end prefetch block command. Unlike add prefetch item, it can only specify one download and relevance substitution is not allowed within the arguments of this command.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
add nohash prefetch item [name=<name>] [size=<size>] url=<url>
```

Where:

- `name` is an optional file name for the download. If no name is specified, it will be automatically determined from the URL.
- `size` is an optional file size.
- `url` is the URL of the file.

The arguments may be in any order, but unrecognized arguments will generate a syntax error.

Note: The downloaded file can be cached only once per action.

Examples

Download and run a file as `insecure.exe` without verifying its hash.

```
begin prefetch block
add nohash prefetch item name=insecure.exe url=http://example.com/some-file
end prefetch block
wait {download path "insecure.exe"}
```

add prefetch item

This command adds a download item to the prefetch queue. This command must reside between a begin prefetch block and an end prefetch block command. This command can specify multiple downloads separated by semicolons.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
add prefetch item [name=<name>] [sha1=<sha1>] [sha256=<sha256>]
size=<size> url=<url> [; ...]
```

Where:

- `name` is an optional file name for the download. If no name is specified, it will be automatically determined from the URL.
- `sha1` is an optional SHA-1 of the file.
- `sha256` is an optional SHA-256 of the file.
- `size` is the size of the file in bytes.
- `url` is the URL of the file.

At least one of `sha1` or `sha256` must be present. To download a file without specifying a hash, use the add nohash prefetch item command.

The arguments may be in any order, and unrecognized arguments will be ignored.

Examples

This example demonstrates a conditional download in a prefetch block. By checking the OS first, only the proper file will be prefetched, potentially saving time and bandwidth.

```
begin prefetch block
if {name of operating system = "Windows 2000"}
add prefetch item {"name=up.exe sha1=12 size=45 url=http://ms.com/
hot2k.exe"}
else
add prefetch item {"name=up.exe sha1=12 size=45 url=http://ms.com/hot.exe"}
endif
end prefetch block
wait {download path "up.exe"}
```

Notes: Relevance substitution is allowed with the arguments of this command. However when substitution is used, the BigFix Server can't cache the download item at action creation time.

Instead of listing the download items in the command line, you can put them in a file (one item per line) and then use a relevance substitution like the following:

```
begin prefetch block
add prefetch item {concatenation ";" of lines of file my-downloads.txt}
```

```
end prefetch block
```

begin prefetch block

This command starts a set of commands to download files.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
begin prefetch block
```

Examples

Download the file `myfile.txt`.

```
begin prefetch block
add prefetch item name=myfile.txt
 sha1=09d24c690168f084287af838008cbceca8215425
size=234 url=http://example.com/myfile.txt
end prefetch block
```

Notes: Only one prefetch command block can be used in an action and it must be closed with an end prefetch block command. Only comments or blank lines are allowed to precede this command. When processing actions with prefetch blocks, download, download as and prefetch are not allowed anywhere in the action script. The download now as command is allowed, but it must be used after the prefetch block. These commands are allowed within the prefetch block:

- if, elseif, else, endif
- parameter
- action parameter query

These commands are only allowed within the prefetch block. They are not allowed outside of it:

- add prefetch item
- add nohash prefetch item
- collect prefetch items
- execute prefetch plug-in

collect prefetch items

After files have been added to the prefetch queue by commands such as add nohash prefetch item and add prefetch item, this command tells the client to download those files and to not continue running the action until the files have been downloaded. This command is typically used to retrieve a download plug-in or a set of files that can be processed by a plug-in. In this case, a file is first added to the prefetch list, collected, and then processed by a subsequent execute prefetch plug-in command, which might create a file containing additional downloads. Each collect prefetch items command is treated as a synchronization point, causing the prefetch processing of the action to wait for the files to download before proceeding. Once the files are available, the action is reprocessed from the beginning. This allows the action to compensate for any files that may have changed due to altered conditions on the machine. The next command in the action will be processed only after the collect prefetch items command is executed and all files in the prefetch list have been downloaded. The end prefetch block command does an automatic collection, ensuring that subsequent action commands will have the necessary files available.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
collect prefetch items
```

Examples

Download the prefetch plugin `myPlugIn.exe` and run it to add more dynamic downloads to be prefetched.

```
begin prefetch block
parameter "ini"="{file "server_bf.ini" of site (value of setting
 "MyCustomSite") of client}"
add prefetch item name=myPlugIn.exe
 sha1=78ed0f73e7e34e0d0882dd453be0c5ac0f0913eb size=1240
url=http://mysite/plugin.exe
// collect the plug-in before continuing:
collect prefetch items
execute prefetch plug-in "{download path "myPlugIn.exe"}" /downloads
 "{parameter "ini"}"
"{download path "urllist"}"
add prefetch item {concatenation " ; " of lines of download file "urllist"}
end prefetch block
```

download

This command downloads a file from a URL. After downloading, the file is saved in a folder named __Download relative to the local folder of the site that issued the download command. The name of the file is derived from the part of the URL after the last slash. If the download fails, the action script terminates.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
download <url>
```

Examples

Download `bfxxxx.exe` from the BigFix site, and save the downloaded file in the default site `__Download` folder.

```
download http://download.bigfix.com/update/bfxxxx.exe
```

Notes: Relevance substitution is **not** performed on the download action command lines. This is because these actions are scanned by other components that deliver the downloads, and these other components run on different machines which do not share those client's evaluation context. This restriction, however, allows BigFix to prefetch downloads through a relay hierarchy to the clients.

download as

This command downloads a file from a URL and allows you to rename it. After downloading, the file is saved in a folder named `__Download` relative to the local folder of the site that issued the `download as` command.

For instance, consider the command:

```
download as intro.txt ftp://ftp.microsoft.com/deskapps/readme.txt
```

The example downloads the `readme.txt` file from the Microsoft site and saves it in the local `__Download` folder as `intro.txt`. If the download fails, the action script terminates.

This command, when accompanied by a continue if with a sha1 value, allows the file to be pre-fetched.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
download as <name> <url>
```

Where `name` is a simple filename, without special characters or path delimiters. If the name violates any of the following rules, the download command will fail:

- Name must be 32 characters or less.
- Name must only be composed of ASCII characters a-z, A-Z, 0-9, -, _, and non-leading periods.

Examples

Download the `prog555.exe` file from the specified folder on the web site, saves the downloaded file to the action site `__Download` folder and renames it to `myprog.exe`.

```
download as myprog.exe http://www.website.com/update/prog555.exe
```

Downloads the specified file, renames it `patch1` and continues only if the size and sha1 are correct.

```
download as patch1 http://www.download.windowsupdate.com/some-update.exe
continue if {(size of it = 813160 and sha1 of it
 ="92c643875dda80022b3ce3f1ad580f62704b754f") of file
"patch1" of folder "__Download"}
```

Notes: Relevance substitution is **not** performed on the download action command lines. This is because these actions are scanned by other components that deliver the downloads, and these other components run on different machines which do not share those client's evaluation context. This restriction, however, allows BigFix to prefetch downloads through a relay hierarchy to the clients.

download now

This command downloads a file from a URL. It is simliar to the download command. However, unlike that command, the client will download directly from the specified URL at that point in the action script without using the relay hierarchy.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
download now <url>
```

Examples

Downloads `bfxxxx.exe` from the BigFix site as soon as the command is executed.

```
download now http://download.bigfix.com/update/bfxxxx.exe
```

Notes: Relevance substitution is **not** performed on the download action command lines. This is because these actions are scanned by other components that deliver the downloads, and these other components run on different machines which do not share those client's evaluation context. This restriction, however, allows BigFix to prefetch downloads through a relay hierarchy to the clients.

download open

This command downloads a file from a URL and then runs ShellExecute on the resulting file.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
download open <url>
```

Examples

Download and save `bfxxxx.exe` to the default site `__Download` folder and execute the program once the download completes.

```
download open http://download.bigfix.com/update/bfxxxx.exe
```

Notes: Relevance substitution is **not** performed on the download action command lines. This is because these actions are scanned by other components that deliver the downloads, and these other components run on different machines which do not share those client's evaluation context. This restriction, however, allows BigFix to prefetch downloads through a relay hierarchy to the clients.

end prefetch block

This command marks the end of a prefetch block. This command must be present whenever the begin prefetch block command is specified. This command automatically performs a collect prefetch items command, meaning that all the files added to the prefetch list will be available when the block is ended.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
end preftech block
```

Notes: See begin prefetch block for more information about prefetch blocks.

execute prefetch plug-in

This command runs an external binary in a prefetch block. This is most commonly used to produce another file containing a set of URLs to be downloaded. This can be used to authenticate or execute downloads. It can also be used to execute custom logic that can create inspectable values for subsequent add prefetch item commands. It is not intended for a lengthy executable and the client will only wait 60 seconds for its completion.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
execute prefetch plug-in <executable-path> [args]
```

Where:

- `executable-path` is the full pathname for the plug-in to execute.
- `args` are arguments passed to the executable

Examples

This example downloads a plug-in that processes the `ini_file` to produce a download manifest.

```
begin prefetch block
```

```
parameter "ini_file"="{file "server_bf.ini" of site (value of setting
 "MyCustomSite") of client}"
add prefetch item name=plugin.exe
 sha1=321381802e1689728e63f25496f8feda98cb3c6e size=1573
url=http://mysite/myplugin.exe
collect prefetch items

// execute the plug-in to produce a manifest from the ini_file:
execute prefetch plug-in "{download path "plugin.exe"}" /downloads
 "{parameter "ini_file"}"
"{download path "manifest"}"
add prefetch item {concatentation " ; " of lines of download file
 "manifest"}
end prefetch block
```

Notes:

The exit code of the executable is important as it informs the client of failure or success, where `0` indicates success and all other exit codes are treated as failures and result in a failed action. For debugging purposes, the exit code is recorded in the client log.

This command is designed for executables that are fast to execute and return promptly. The BigFix client will only wait 60 seconds for the plugin to complete. After 60 seconds, the client will log a message and disable the command. When it is disabled, any actions that use this command will not execute until after the client has been restarted.

In general it is expected that the command will complete much faster -- if it takes longer than 2 seconds to execute, the client will log an appropriate message.

Relevance substitution can be used to specify the pathname.

prefetch

This command allows a file to be downloaded before the action begins. You do not need a matching continue if statement for the file to be downloaded and checked in advance. This

command is preferred over the download command. To easily create prefetch commands, the make-prefetch utility can be used.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
prefetch <name> sha1:<sha1> size:<size> <url> [sha256:<sha256>]
```

Where:

- `name` is the file name for the download.
- `sha1` is the SHA-1 of the file.
- `size` is the size of the file in bytes.
- `url` is the url of the file.
- `sha256` is an optional SHA-256 of the file.

The `name` must be a simple filename, without special characters or path delimiters. If the name violates any of the following rules, the download command will fail:

- Name must be 32 characters or less.
- Name must only be composed of ASCII characters a-z, A-Z, 0-9, -, _, and non-leading periods.

Examples

Prefetch a picture of Hodor.

```
prefetch hodor.jpg sha1:ce842e0af799f2ba476511c8fbfdc3bf89612dd0
size:57656 http://i.imgur.com/YAUeUOG.jpg sha256:74f69205a016a3896290eae0
3627e15e8dfeba812a631b5e0afca140722a322b
```

Prefetch and run a different patch depending on whether the operating system is Windows XP.

```
if {name of operating system = "WinXP"}
prefetch patch.exe sha1:92c643875dda80022b3ce3f1ad580f62704b754f
 size:813160
http://www.download.windowsupdate.com/msdownload
/update/v3-19990518/cabpool/
q307869_f323efa52f460ea1e5f4201b011c071ea5b95110.exe
else
prefetch patch.exe sha1:c964d4fd345b6e5fd73c2235ec75079b34e9b3d2
 size:845416
http://www.download.windowsupdate.com/msdownload/update/v3-19990518/
cabpool/q310507_2f3c5854999b7c58272a661d30743abca15caf5c.exe
endif
wait __Download\patch.exe
```

Execution Commands

These commands allow you to run external commands, and to change the behavior of how those commands are run.

action launch preference low-priority

When this command is run, subsequent action commands that launch programs will do so with lower priority than normal. This will help to mitigate the impact of large patches or service pack upgrades. Low-priority preference only affects the launch priority of applications launched from the current action. This preference is maintained until the action completes or the client executes the action launch preference normal-priority command.

Version	Platforms
8.0.584.0	Windows

Syntax

```
action launch preference low-priority
```

Examples

This example lowers the launch priority before running `background_app` so that it will not dominate the system when it executes. It then sets the priority level back to normal.

```
action launch preference low-priority
run "{pathname of regapp "background_app.exe"}"
action launch preference normal-priority
```

Notes: This command is Windows-only. It will cause an action script to terminate on a Unix agent.

action launch preference normal-priority

When this command is executed, subsequent action commands that launch programs will do so with normal-priority. This statement is only needed to return the priority to normal after an action launch preference low-priority command.

Version	Platforms
8.0.584.0	Windows

Syntax

```
action launch preference normal-priority
```

Examples

This example lowers the launch priority before running `background_app`, then returns the priority to normal for subsequent launch statements.

```
action launch preference low-priority
run "{pathname of regapp "background_app.exe"}"
action launch preference normal-priority
```

Notes: This command is Windows-only. It will cause an action script to terminate on a Unix agent.

action uses wow64 redirection

By default and for ensuring the backward compatibility with 32-bit systems, the BigFix Agent operates in a 32-bit context for running actions. This means that the paths to your files are automatically translated by Windows into the 32-bit equivalent paths. This is the so-called wow64 redirection. You can prevent the wow64 redirection when running actions by using the action uses wow64 redirection command with the false option. The to prevent wow64 redirection and . If you want the agent to run actions that avoid the wow64 redirection, then you can run the command action uses wow64 redirection false, which will avoid the redirection. The action uses wow64 redirection command affects the behavior of the following action commands: dos, delete, copy, move, open, run, wait (and their variants such as waithidden).

Version	Platforms
8.0.584.0	Windows

Syntax

```
action uses wow64 redirection <true|false>
```

Examples

To run the 64-bit version of notepad.exe in the system path, you can run the following command:

```
action uses wow64 redirection {not x64 of operating system}
run notepad.exe
```

Notes: This command is Windows-only. It will cause an action script to terminate on a Unix agent.

action uses file encoding

The action uses file encoding command affects the behavior of the appendfile and createfile until commands. If you do not use the action uses file encoding command, the appendfile and createfile until commands create files in the local client encoding. The encoding might be any name that the International Components for Unicode (ICU)

can recognize, such as ISO-8859-1, Shift_JIS, and UTF-8. If any of UTF encodings (UTF-8, UTF-16, or UTF-32) is specified as the value of encoding, the file to be created will have a BOM (Byte Order Mark) at the head of it. But, in case that the client's local encoding is UTF-8 and no encoding is explicitly specified in an action, files to be created with the action will be written in UTF-8 without BOM. This behavior is the same with the existing version of the "appendfile" and "createfile until" commands, so we should keep the same behavior unless any encoding is specified so that the existing actions will work as before. To suppress adding any BOM, you can pass an optional suboption "NoBOM" (case-insensitive) following the value of encoding. The "NoBOM" suboption is effective only with any UTF encodings (UTF-8, UTF-16, and UTF-32), and it will be ignored if it is passed with any other encoding name. After created, the file objects can be used as regular file objects and you can apply any operations applicable to text files. To turn off the encoding change and reuse the local encoding, you can set the encoding name to local.

VersionPlatforms

9.5.7 All

Syntax

```
action uses file encoding encoding [ NoBOM ]
```

Examples

To create a file using the UTF-8 encoding without a BOM, you can run the following command:

```
action uses file encoding UTF-8 NoBOM
```

To revert using the local encoding, run the following command:

```
action uses file encoding local
```

On non-Windows platforms:

```
delete "{(client folder of current site as string) & "/__appendfile"}"
action uses file encoding UTF-8 noBOM
appendfile Hello world !!
delete /tmp/encode_test.txt
```

```
move __appendfile /tmp/encode_test.txt
```

dos

Runs a Windows command. If the command fails, the action script that contains it is terminated.

Version	Platforms
8.0.584.0	Windows

Syntax

```
dos <command-line>
```

Examples

Delete an empty directory from a temporary folder in the windows directory:

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

Run `scandisk.exe` on the `E:` drive:

```
dos scandisk.exe e:
```

Notes: This command is Windows-only. It will cause an action script to terminate on a Unix agent. On a Windows system, this has the same effect as issuing a system statement from the Windows API. It is also the same as typing the command line to a command prompt. The dos command uses the PATH environment variable to locate the command on the user's hard drive. As with any other dos command, for other locations you must specify a complete pathname. Be sure to use quotes if you have spaces in the filenames.

override

The override command provides the ability to customize certain commands and add multiple variations to existing commands. This powerful compound command allows you to create your own custom combination command similar to the existing commands waitdetached or runhidden. To add constraints to an existing command, you add predefined keyword/value pairs within the body of the command.

Warning: Do not launch long run programs directly from the __Download folder using any of the following commands: run, rundetached, runhidden, override with completion=none, or override with timeout, disposition=abandon. Instead, add an action to copy the programs to a location different from the __Download folder and launch the programs from there. This is necessary because, if a file in the __Download folder is invoked from any of these action script commands, the launched program locks the file until it ends and, if the launched program runs for a very long time or hangs, the Agent cannot process the next action for the same site context because it cannot clear the __Download folder.

Version	Platforms
8.2.531.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows, Debian, Ubuntu

Syntax

```
override <cmd>
<keyword>=<value>
<keyword>=<value>
<cmd> <rest of command-line>
```

Keywords

The keywords may be specified in any order, but there must be only one per line. White-space is not needed around the equal sign = and is ignored.

Keywords are case-insensitive, and the values can be enclosed in {curly brackets} for Relevance substitution.

If duplicate keywords are listed, the last value will be used. The entire command fails if any of the keywords or values are invalid. Platform-specific keywords that are not meaningful on a given platform will be silently ignored.

The action command overrides timeout_seconds and disposition only modify the behavior of the `wait` and `waithidden` action commands.

Completion

Default value: `none` for run, `process` for wait.

- `Completion=none` acts the same as the current run command variants.

- `Completion=process` acts the same as the current wait command variants. When this value is used for run, the command result it is not displayed.
- `Completion=job` on Windows makes use of Windows Job Objects which imposes some limitations on the target process and some potential failure points for the command.

Priority (Windows Only)

Default value: `normal`

- `Priority=normal` acts the same as the action launch preference normal-priority command.
- `Priority=low` acts the same as the action launch preference low-priority command.

Hidden (Windows Only)

Default value: `false`

- `Hidden=true` applies the `SW_HIDE` attribute to the process as is done with the runhidden and waithidden commands.
- `Hidden=false` removes the `SW_HIDE` attribute from the process.

Detached (Windows Only)

Default value: `false`

- `Detached=true` creates the process using the detach method as is done in the rundetached and waitdetached commands.
- `Detached=false` creates the process using the normal method.

RunAs

Use this keyword to specify the user and the context to use when running the command specified in the action. Default value: `agent`

- `RunAs=agent` applies the same process ownership characteristics as the current wait and run commands. The user and the context are the same as those used to with the current wait and run commands.
- `RunAs=currentuser` mimics `RunAsCurrentUser.exe` on Windows, using the same logic to identify the current user and similar code to create the process with an environment block sourced by the userToken. In case of multiple logged-on users, the BES agent chooses the console session if active, or the first logged-on user returned from the operating system.

  Note: On UNIX and Linux the environment variables are not applied with the exception of required Xauthority variables. On such platforms a call is made to setuid to the id of the user identified as the current user for the XBESClientUI. This is a very specific and platform dependent scenario which requires the user to be logged on at the local console and running X Windows. In the case of an Offer, there is no relationship between the user who has accepted the Offer and the current user identified by the BES agent at the time of action execution.
- `RunAs=localuser` specifies a user who can be different from the logged on user.

  Specify the mandatory option `user` in one of these two formats: `user=<username>`, or `user={relevance to describe the username}` to allow a parametrized input.

  This keyword requires the BigFix Agent to run successfully, for this reason it does not work when run from the Fixlet Debugger.

  On Windows systems you can specify any local or domain account. If you use the keyword `Completion=process` or `Completion=job`, there is no need for the specified user to be logged on the system in advance. If you use the keyword `Completion=none`, the user must be logged on the system in advance and must have a registry hive.

  On other operating systems, the specified user must be either local or listed in local accounts, in other words:
  - On Unix or Linux the user must be specified in the */etc/passwd* file.
  - On Mac the user must be one of the locally defined users.

The following considerations about the `RunAs=localuser` keyword apply to users defined on Windows systems only:

- You can specify a domain user by using one of the following formats:
  - "user@domain" where domain is an active directory domain, for example "john@tem.test.com".
  - "domain\user" where domain is specified in the short domain name notation, for example "TEM\john".

    Note: The action runs even if the specified domain user has never logged on the target system before then.
- You can specify the option `password` as follows:
  - `password=required` if specified, a Take Action Dialog requiring to enter the user's password is displayed on the Console. That password is then passed to the agent as a SecureParameter.

    Note: Only one password can be passed to the action using the override command. An action with more than one override command, with different users specified, fails unless the specified users use the same password. To bypass this constraint, you might want to create different Fixlets or tasks, each one with an action containing one of the override commands to run.
  - `password=impersonate` if the agent must search for a session running with the user specified in the `user` option, and run the command in that session.
  - `password=system` to run the command the with the Local System account and without an user context. The command requires the specified user to be logged on when the `override` runs on the system. Any UI will be displayed in the session of the specified user.

    Note: Use the `asadmin` option if you want the command to write to HKEY_CURRENT_USER registry hive.

    Note: On other operating systems, the option `password` is ignored because the agent runs with root authority.

You can use the option `asadmin` as follows:

- `asadmin=true` to run the specified command in the context of the specified user as if user is a member of builtin Administrators group. In this case you must:
  - Specify `password=required`.
  - Omit the `targetuser` keyword.
- `asadmin=interactive` to run the specified command in the context of the user specified in the `user` keyword as if that user were a member of builtin Administrators group. The following rules apply if you use this value:
  - If you use the `targetuser` keyword, the UI launched by the command is displayed in the session of the user specified with `targetuser`. The command fails if the user specified with the `targetuser` keyword is not logged on when the override command runs.
  - You must specify the keyword `password=required`, when using an existing account, or the `password="password"` keyword, if you want to use a temporary user and you have to specify the user's password in the action in clear text. If you use the `password="password"` keyword, specify the actual password surrounded with double quotes.

timeout_seconds

Default value: 0

- timeout_seconds=*positive integer* makes the client wait the specified number of seconds during a wait or waithidden command before the action script continues without waiting for the completion of the command's process. The supported values for the timeout are all positive integers to the maximum supported by the computer architecture.
- timeout_seconds=0 is the default value, and makes the wait or waithidden commands act as if the timeout_seconds override were not set.

disposition

Default value: `abandon`

- disposition tells the client what to do with the process under the wait or waithidden command once the timeout is finished.
- disposition=terminate tells the client to kill the wait and waithidden command's process when the timeout is reached. Killing the process can have negative consequences, and should be used with extreme caution.
- disposition=abandon is the default value, and tells the client to disassociate the wait or waithidden command's process from the remainder of the actions.

Limitations on Completion=job

Windows

To exercise the most flexible job control over a process, the override command allows the process to selectively break child processes away from the job. This allows the process to do its own job control management, but removes any of its broken out children from the job object.

In those limited cases where the launched process is responsible for its own job control, it is assumed that a member of the job will remain running until all of its child processes complete. This is not a guarantee, however, and there may be situations where this is not the case. In those cases, the action completes even though the child processes are still running.

UNIX/Linux

On UNIX/Linux platforms session IDs are used to manage job processes. Session IDs take on the value of the process id of the session leader (the process you want to launch). The client waits for the leader process to end, as in the `Completion=process` case, then begins a cycle of a half-second of sleep followed by enumerating processes looking for anything with a session id matching the job leader's process id. When no more of these processes exist, the job is complete and the command finishes.

The exit code returned with the command is always that of the leader process, not the last process to complete.

Examples

On Windows platforms

This example provides the same functionality as `waithidden notepad.exe`:

```
override wait
hidden=true
wait notepad.exe
```

This example shows how you might run a patch as a hidden process by the current user, waiting for completion of the job before continuing the action script:

```
override wait
completion=job
hidden=true
runas=currentuser
wait __Download\patch.exe arg1 arg2 arg3
```

This example shows how you might run a maintenance application, but kill the maintenance process if it isn't finished by the time 1 hour has passed:

```
override wait
timeout_seconds=3600
disposition=terminate
wait "__Example\maintenance.exe" arg1 arg2 arg3
```

This example shows how you might install a software application on a Windows machine using the context of a domain user who doesn't belong to the Administraor group. A Take Action Dialog, asking for the user's password, will appear on the console . The password will be passed to the agent as a SecureParameter:

```
override wait
        runas=localuser
    asadmin=true
    user=TEM\User1
    password=required
wait c:\IMAGE\SWD\application.exe /SILENT
```

On non-Windows platforms

```
override wait
completion=job
wait tar --directory=/tmp -zxvf __Download/myFile.tgz
```

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 wish to wait for one program to finish before starting another one, use the wait command.

Warning: Do not launch long run programs directly from the __Download folder using any of these commands: run, rundetached, runhidden, override with completion=none, or override with timeout, disposition=abandon. Instead, add an action to copy the programs to a location different from the __Download folder and launch the programs from there.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
run <command-line>
```

Examples

These examples show how you might run a script and pass it some arguments. Quotes around the command line are recommended, and necessary if there are spaces in file names.

On Windows platforms

```
run "{pathname of regapp "wordpad.exe"}"
run "c:\winnt\ftp.exe" ftp.mycorp.net
```

```
run wscript /e:vbs x.vbs arg1 arg2
```

On non-Windows platforms

```
run sudo touch "/tmp/example.txt"
```

Notes: On a Windows computer, this command has the same effect as calling the CreateProcess function with <command-line>. This is also the same as using <command-line> in the Windows Run dialog. See the Windows documentation on CreateProcess for a discussion of the method used to locate the executable from a <command-line>.

rundetached

This command is used to prevent pop-up DOS windows when waiting for a program to complete. It's the same as the run command, but the process created doesn't access the parent's console, inhibiting the distracting DOS window. This command modifies the run command by setting the DETACHED_PROCESS flag when calling CreateProcess on Windows machines. By default, a created process inherits its parent's console. When detached, this behavior is inhibited. This gives the new process some more control over how it may interact with the user. This command should not be used for running interactive programs. If this is done, the interactive program will not be able to show its user interface and may appear to be hung. This command is provided strictly for running programs that do not display a user interface.

Warning: Do not launch long run programs directly from the __Download folder using any of these commands: run, rundetached, runhidden, override with completion=none, or override with timeout, disposition=abandon. Instead, add an action to copy the programs to a location different from the __Download folder and launch the programs from there.

Version	Platforms
8.0.584.0	Windows

Syntax

```
rundetached <command-line>
```

Examples

These examples show how you might run a program and pass it some arguments. Quotes around the command line are recommended, and necessary if there are spaces in file names.

```
rundetached "{pathname of regapp "background_app.exe"}"
rundetached "c:\winnt\ftp.exe" ftp.filesite.net
```

Notes: This command is Windows-only. It will cause an action script to terminate on a Unix agent. On a Windows computer, this command has the same effect as calling the CreateProcess function with <command-line>. This is also the same as using <command-line> in the Windows Run dialog. See the Windows documentation on CreateProcess for a discussion of the method used to locate the executable from a <command-line>.

runhidden

This command uses CreateProcess to launch a command in a hidden window. It hides the window by setting the STARTUPINFO dwFlags to STARTF_USESHOWWINDOW and setting wShowWindow to SW_HIDE. The process that is created may modify that flag to subsequently show the window again. After launching, the following action command line is immediately executed. To wait for the launch to complete before continuing the action, use the waithidden command.

Warning: Do not launch long run programs directly from the __Download folder using any of these commands: run, rundetached, runhidden, override with completion=none, or override with timeout, disposition=abandon. Instead, add an action to copy the programs to a location different from the __Download folder and launch the programs from there.

Version	Platforms
8.0.584.0	Windows

Syntax

```
runhidden <command-line>
```

Examples

These examples show how you might run a script in a hidden window and pass it some arguments. Quotes around the command line are recommended, and necessary if there are spaces in the file names.

```
runhidden "{pathname of regapp "wordpad.exe"}"
runhidden "c:\winnt\ftp.exe" ftp.mycorp.net
runhidden wscript /e:vbs x.vbs arg1 arg2
```

Notes: This command is Windows-only. It will cause an action script to terminate on a Unix agent. If the launched process requires user input, it will wait for it with its window hidden, unless the command explicitly shows its window. On a Windows computer, this command has the same effect as calling the CreateProcess function with <command-line>. This is also the same as using <command-line> in the Windows Run dialog. See the Windows documentation on CreateProcess for a discussion of the method used to locate the executable from a <command-line>.

script

This command executes an external script with the given name. This can be used to run a script created for a scripting language like JavaScript or Visual Basic.

The action 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 is not executed until the specified script terminates.

Version	Platforms
8.0.584.0	Windows

Syntax

```
script <script-name>
```

Examples

Run the Visual Basic script `attrib.vbs`.

```
script attrib.vbs
```

Notes: This command is Windows-only. It will cause an action script to terminate on a Unix agent. On a Windows computer, this command has the same effect as issuing a wscript "scriptName" statement from Windows using wscript.exe, 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.

script64

This command uses the same syntax as the script command, but places a call to Wow64DisableWow64FsRedirection before executing the script. This allows you to issue a native 64-bit script command, bypassing Windows 32-bit environment built on top of 64-bit processors.

This command executes an external script with the given name. This can be used to run a script created for a scripting language like JavaScript or Visual Basic.

The action containing the `script64` keyword will terminate if the appropriate scripting engine is not installed or if the script cannot be executed. The next line of the action is not executed until the specified script terminates.

Version	Platforms
8.0.584.0	Windows

Syntax

```
script64 <script-name>
```

Examples

Run the Visual Basic script `attrib.vbs` in native 64-bit mode.

```
script64 attrib.vbs
```

Notes: This command is Windows-only. It will cause an action script to terminate on a Unix agent. On a Windows computer, this command has the same effect as calling Wow64DisableWow64FsRedirection, then issuing a wscript "scriptName" statement from Windows using wscript.exe, and then waiting for completion.

wait

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

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
wait <command-line>
```

Examples

Runs the `scandskw.exe` program and waits for the program to complete before continuing with the action script. The use of quotes is recommended practice, and necessary if there are spaces in the file name.

```
wait "scandskw.exe"
```

On non-Windows platforms:

```
wait chmod 555 inventory.sh
```

Notes: On a Windows computer, this command has the same effect as calling the CreateProcess function with <command-line>. This is also the same as using <command-line> in the Windows Run dialog. See the Windows documentation on CreateProcess for a discussion of the method used to locate the executable from a <command-line>. The wait command has two available override keywords: timeout_seconds, and disposition. See the override documentation for details.

waitdetached

This command 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, inhibiting the distracting DOS window.

This command modifies the wait command by setting the DETACHED_PROCESS flag when calling CreateProcess on Windows machines. By default, a created process inherits its parent's console. When detached, this behavior is inhibited. This gives the new process some more control over how it may interact with the user.

This command should not be used for running interactive programs. If this is done, the interactive program will not be able to show its user interface and may appear to be hung. This command is provided strictly for running programs that do not display a user interface.

Version	Platforms
8.0.584.0	Windows

Syntax

```
waitdetached <command-line>
```

Examples

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

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

Notes: This command is Windows-only. It will cause an action script to terminate on a Unix agent. On a Windows computer, this command has the same effect as calling the CreateProcess function with <command-line>. This is also the same as using <command-line> in the Windows Run dialog. See the Windows documentation on CreateProcess for a discussion of the method used to locate the executable from a <command-line>.

waithidden

This command uses CreateProcess to launch a command in a hidden window. It hides the window by setting the STARTUPINFO`dwFlags` to `STARTF_USESHOWWINDOW` and setting `wShowWindow` to `SW_HIDE`. The process that is created may modify that flag to subsequently show the window again.

This command waits for the completion of the process before continuing with subsequent action commands.

Version	Platforms
8.0.584.0	Windows

Syntax

```
waithidden <command-line>
```

Examples

These examples show how you might run a script in a hidden window and pass it some arguments. Quotes around the command line are recommended, and necessary if there are spaces in the file names.

```
waithidden "{pathname of regapp "notepad.exe"}"
waithidden "c:\winnt\ftp.exe" ftp.myurl.net
waithidden wscript /e:vbs x.vbs arg1 arg2
```

Notes: This command is Windows-only. It will cause an action script to terminate on a Unix agent. If the launched process requires user input, it will wait for it with its window hidden, unless the command explicitly shows its window. On a Windows computer, this command has the same effect as calling the CreateProcess function with <command-line>. This is also the same as using <command-line> in the Windows Run dialog. See the Windows documentation on CreateProcess for a discussion of the method used to locate the executable from a <command-line>. The waithidden command has override keywords. See the override documentation for details.

Flow Control Commands

These commands allow you to add conditional logic to your action script.

action may require restart

This command looks at the system for signs that a restart is needed. If it seems that a restart is needed, the action completion status will be set to Pending Restart until a restart occurs. If the optional name argument is given and a restart is needed, then the client will mark name as needing a restart. The pending restart inspector can then be used to see which actions require a restart.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
action may require restart [name]
```

Examples

This command is commonly used with patches from Windows Update that might require a restart. For example:

```
prefetch WindowsXP-KB2914368-x86-ENU.exe
 sha1:1d9a306f9e5dd564c8ffdcdb8717c4ae2588db3d size:530672
http://download.microsoft.com/download/B/0/D/
B0D762B1-1CF4-4377-8149-0FB18167A023/WindowsXP-
KB2914368-x86-ENU.exe
 sha256:6a8e3034478704c7701e2e2279811e278eec45cc218f50c8ab0701a6b732afc4

waithidden __Download\WindowsXP-KB2914368-x86-ENU.exe /quiet /norestart

action may require restart "1d9a306f9e5dd564c8ffdcdb8717c4ae2588db3d"
```

If this action requires a restart, then this relevance will return `True`:

```
pending restart "1d9a306f9e5dd564c8ffdcdb8717c4ae2588db3d"
```

action parameter query

This command allows you to prompt the user that creates the action for the value of a parameter. This is typically used to prompt for setting values, file locations, or service names. Parameter names may include blanks, and are case sensitive. The parameter name, description, and value must each be enclosed inside double quotation marks ".

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

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

Examples

Ask the user for the value of `InstallationPoint` when this action is taken. When the script is run, use the value of `InstallationPoint` as an argument to `MyTool.exe`:

```
action parameter query "InstallationPoint" with description "Please enter
 the location of the
shared installation point:"
run MyTool.exe {parameter "InstallationPoint"}
```

Create a parameter that can turn 'Tips' on or off:

```
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"}"
```

Notes: While the action is executing, you can retrieve the action parameter value entered by the user with the parameter inspector. For example, in your action you could use relevance substitution:

```
run "{parameter "parameter name"}"
```

Relevance substitution is **not** performed on the action parameter query command line itself. This is because the command is interpreted in the BigFix Console before the action is sent out.

action requires login

This command informs the client that the current action will not be completed until the computer is restarted and an administrator logs in. Once this commnand runs, the pending login inspector will return true.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
action requires login
```

Notes: This command is ignored by Unix clients.

action requires restart

This command will place the action in a Pending Restart state until the computer is restarted. If the optional name is specified, then the pending restart inspector will report that name requires a restart.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
action requires restart [name]
```

Examples

Require a restart before the action is reported as completed.

```
run patch.exe
```

```
action requires restart "my-awesome-patch"
```

After running this, the action will be in a *Pending Restart* state and the following relevance will return `True`:

```
pending restart "my-awesome-patch"
```

continue if

This command stops running an action script if a relevance expression evaluates to `False`.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
continue if <condition>
```

Examples

Download the file `dun40.exe` if the operating system is Windows 2000.

```
continue if {name of operating system = "Win2k"}
download now http://www.example.com/downloads/win98/dun40.exe
```

exit

This command terminates the action and sets the action exit code. Relevance substitution can be used to set the exit code.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
exit <code>
```

Examples

Exit the action early if `foo.exe` returns a non-zero exit code.

```
wait "foo.exe"
parameter "error" = "{exit code of action}"
if {parameter "error" != "0"}
  exit {parameter "error"}
endif
// continue processing
```

Notes: This is one of the four script commands that can change the action exit code. The other commands that can change the exit code are:

- wait
- waithidden
- waitdetached

For actions of type sh the exit code of the script is collected into the inspector value when the client finishes processing the shell script. Exit codes from Unix shell scripts are written to the client log.

if, elseif, else, endif

The `if`, `elseif`, `else`, and `endif` commands allow conditional execution of commands.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
if {expression1}
  statement1
  statement2
  ...
elseif {expression2}
  statement3
```

```
  statement4
  ...
else
  statement5
  statement6
  ...
endif
```

Examples

Prefetch and run a different file depending on the operating system.

```
if {name of operating system = "WinME"}
prefetch patch1.exe sha1:e6dd60e1e2d4d25354b339ea893f6511581276fd
 size:4389760
http://example.com/winme.exe
wait __Download\patch1.exe
elseif {name of operating system = "WinXP"}
prefetch patch2.exe sha1:92c643875dda80022b3ce3f1ad580f62704b754f
 size:813160
http://example.com/winxp.exe
wait __Download\patch2.exe
else
prefetch patch3.exe sha1:c964d4fd345b6e5fd73c2235ec75079b34e9b3d2
 size:845416
http://example.com/win7.exe
wait __Download\patch3.exe
endif
```

Notes

The client parses actions before it actually executes them, looking for downloads to prefetch.

If the prefetching process doesn't parse properly, an action syntax error will be returned and the action will not run. This can be problematic if you are creating actions that work in

multiple environments where only one branch of an if statement may parse correctly. For instance, you might want to load files that are unique to specific platforms. A script like this would seem to do the trick:

```
if {not exists key "foo" of registry}
prefetch windows_file ...
else if {not exists package "bar" of rpm}
prefetch unix_file ...
endif
```

Here a Windows registry key triggers the first prefetch, while a Unix package triggers the second. The problem is that the registry inspector will fail on Unix systems, and the package inspector will fail on Windows, causing the prefetch parser to throw an error in both cases.

The answer here is to use cross-platform inspectors to make sure the wrong blocks are not evaluated:

```
if {name of operating system starts with "Win"}
if {not exists key "foo" of registry}
prefetch windows_file ...
endif
else if {name of operating system starts with "Redhat"}
if {not exists package "bar" of rpm}
prefetch unix_file ...
endif
endif
```

By checking first for the proper operating system, you can avoid this type of prefetch parse error.

However, sometimes there may be no way to avoid a potential error. For instance, an action may create and access a file that doesn't yet exist in the prefetch phase:

```
wait chkntfs c: > c:\output.txt
if {line 2 of file "c:\output.txt" as lowercase contains "not dirty"}
regset "HKLM\Software\MyCompany\" "Last NTFS Check"="OK"
```

```
else
regset "HKLM\Software\MyCompany\" "Last NTFS Check"="FAIL"
endif
```

In this Windows example, the output file doesn't exist until the script is actually executed. The prefetch parser will notice that the file doesn't exist when it checks for its contents. It will then throw an error and terminate the action. However, you can adjust the if-condition to allow the prefetch pass to succeed. One technique is to use the active of action expression which always returns False during the prefetch pass. You can use this to avoid the problematic block during the pre-parse:

```
wait chkntfs c: > c:\output.txt
if {not active of action OR (line 2 of file "c:\output.txt" as lowercase
 contains "not dirty")}
regset "HKLM\Software\MyCompany\" "Last NTFS Check"="OK"
else
regset "HKLM\Software\MyCompany\" "Last NTFS Check"="FAIL"
endif
```

By checking first to see whether the action is being pre-parsed or executed, you get a successful prefetch pass and the desired behavior when the action is running.

parameter

This command can be used to create variables in the action script. After setting a variable, the value can be accessed using the parameter inspector. A parameter can only have one value. Trying to set an existing parameter to a different value will result in an error.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
parameter "<name>"="<value>"
```

Note that both the name and the value of the parameter must be inside quotes.

Examples

Defines a parameter named `loc` that contains the pathname of the `tmp` folder, and uses it to write configuration information to a file.

```
parameter "loc" = "{pathname of folder (value of variable "tmp" of
 environment)}"
createfile until end
Operating system = {name of operating system}
Processor count = {number of processors}
end
delete "{parameter "loc"}\config.txt"
copy __createfile "{parameter "loc"}\config.txt"
```

Notes: The saved parameter value is always a string. If there is any error in evaluating the relevance expression to create the parameter, then the parameter will not be set. If the relevance expression results in multiple values then, then the command fails.

pause while

This command pauses action script evalulation while a relevance expression is `True`. It will continue and execute the next command as soon as the expression evaluates to `False` or the expression fails to evaluate.

Use relevance substitution to define the condition.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
pause while <condition>
```

The `condition` should be a relevance expression.

Examples

Pause action script evaluation while `updater.exe` is running.

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

Pause action script evaluation while the file `C:\result.txt` does not exist.

```
pause while {not exists file "C:\result.txt"}
```

File Commands

These commands allow you to copy, move, and delete files.

appendfile

This command creates a text file named `__appendfile` in the site directory. Each time you invoke the command, it appends the specified text to the end of the file. This command may be useful for creating diagnostic files or scripts.

On a typical Windows install this file will be created in:

```
C:\Program Files (x86)\BigFix Enterprise\BES Client\__BESData\<site>
```

The `__appendfile` is automatically deleted before the action script starts running.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
appendfile <text>
```

Examples

This example records information about the OS and moves that information to `C:\info.txt`.

```
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
move __appendfile C:\info.txt
```

This example records the name and the free space available for all the drives on the client computer.

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

Notes

Use the `appendfile` command as part of an action that builds a script which is subsequently passed to a script interpreter. For example, you can use the following syntax to create an `.ini` file:

```
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
```

This same technique can be used to build `.bat` files, `.cmd` files, Visual Basic scripts, bash shell scripts, etc.

archive now

This command invokes the Archive Manager. If the `Archive Operating Mode` is set to manual, this command will trigger archiving and uploading of the configured set of files. To set the appropriate archive mode to manual, use this setting:

```
_BESClient_ArchiveManager_OperatingMode = 2
```

The `archive now` command will fail if the operating mode is not set to manual. It will also fail if an existing archive is currently being uploaded.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
archive now
```

Examples

This command initiates archiving and uploading of the configured set of files.

```
archive now
```

copy

This command copies the source file to the named destination file. The `copy` command fails if the destination already exists or if the copy fails for any other reason such as when the destination file is busy.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
copy <source> <destination>
```

Examples

This command copies the `win.com` file to the `bigsoftware` folder.

```
copy "{name of drive of windows folder}\win.com" "{name of drive of windows
folder}\bigsoftware\win.com"
```

This pair of commmands deletes the target file if it exists before it performs the copy action.

```
delete "c:\windows\system\windir.dll"
copy " __Download\windir.dll" "c:\windows\system\windir.dll"
```

This command copies the file `/test/my-file.txt` to a file named `/temp/###` in UTF-8 encoding under the site context.

```
binary name copy {"/test/my-file.txt" as binary_string as hexadecimal}
{("/temp/" as binary_string as hexadecimal) & "e3838fe383ade383bc"}
```

This command copies the file `/test/my-file.txt` into `/var/opt/BESClient/__BESData/CustomSite_Fabio/###` in UTF-8 encoding.

```
binary name copy {"/test/my-file.txt" as binary_string as hexadecimal}
 e3838fe383ade383bc
```

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

createfile until

This command creates a text file named `__createfile` in the site directory. It allows you to fill a file with a series of statements up to a terminating string. This is similar to the appendfile command and it works like a here document.

This command is typically used to create a config file or a script.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
createfile until delimiter
line 1
line 2
...
delimiter
```

Examples

This example copies some system information to the file `C:\info.txt`:

```
createfile until end
Operating system = {name of operating system}
Processor count = {number of processors}
end
delete C:\info.txt
copy __createfile C:\info.txt
```

delete

This command deletes a file. Any 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.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
delete <filename>
```

Examples

Delete the `module.dll` file.

```
delete "c:\program files\bigsoftware\module.dll"
```

Delete the `win.com` file.

```
delete "{name of drive of windows folder}\win.com"
```

Delete file `/temp/###` in UTF-8 encoding.

```
binary name delete {("/temp/" as binary_string as hexadecimal) &
 "e3838fe383ade383bc"}
```

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

extract

This command extracts files from the specified archive in the download folder `__Download` and leaves the results in the same folder.

An archive file is similar to a compressed tar file. BigFix uses a tool called BFArchive to construct the archive.

This can be useful for copying an entire directory to a computer, which is often required by installers that contain multiple files along with a setup executable.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
extract <archive-file>
```

Examples

Extract files of InstallMyApp in the `__Download` folder, places the results back in the `__Download` folder and deletes the original InstallMyApp file.

```
extract InstallMyApp
```

folder create

This command creates a directory. It will fail if the folder cannot be created.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
folder create <path>
```

Examples

Create the `C:\test` folder.

```
folder create "C:\test"
```

Under the current folder create folder "/\□" in UTF-8 encoding.

```
binary name folder create e3838fe383ad
```

Under /tmp/ folder create folder "/\□—" in UTF-8 encoding.

```
binary name folder create {("/temp/" as binary_string as hexadecimal) &
 "e3838fe383ade383bc"}
```

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

folder delete

This command deletes a directory. It will recursively delete all contained files and folders. The command fails if the directory cannot be deleted. However, if the directory does not exist, the command succeeds.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
folder delete <path>
```

Examples

Delete the directory `C:\test`.

```
folder delete "C:\test"
```

Remove the folder `/temp/###` in UTF-8 encoding.

```
binary name folder delete {("/temp/" as binary_string as hexadecimal) &
 "e3838fe383ade383bc"}
```

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

move

This command moves the source file to the named destination file. This command can be used to rename a file. The `move` command fails if the destination already exists, if the source file doesn't exist, or if the move fails for any other reason.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
move <source> <destination>
```

Examples

This command moves and renames the `mod.dll` file. Note that quotes are necessary for file names and folder names with spaces in them.

```
move "c:\program files\bigsoftware\module.dll" "c:\temp\mod.dll"
```

This script first deletes the file, then moves it back in place from another location.

```
delete "c:\updates\q312456.exe"
move "__Download\q312456.exe" "c:\updates\q312456.exe"
```

This command moves the file `/test/my-file`.txt into the file `/temp/###` in UTF-8 encoding.

```
binary name move {"/test/my-file.txt" as binary_string as hexadecimal}
{("/temp/" as binary_string as hexadecimal) & "e3838fe383ade383bc"}
```

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

utility

This command can be used to place commonly used programs into a special cache.

The client maintains two disk caches: one for utility programs and another for action payloads. Files arriving in the action payload cache will not push files out of the utilities cache and vice versa.

The client uses the sha1 value of an action download to locate any matching utility that already exists on the client.

An action-specific folder is created to contain downloads as they are pre- fetched. Existing files that match the sha1 values don't need to be downloaded again. Any other files will be prefetched from the parent relay. When all the downloads are available on the client, the files will be moved from the action- specific folder to the `__Download` folder of the action site and the action will be started.

When the action completes, any files left in the `__Download` folder that were pre-fetched with sha1 will be candidates for utility caching. These files will have their sha1 values re-computed and any files with matching sha1 values can be moved into the utility cache.

A least-recently used scheme is used to keep the cache within its size limits. For short intervals only, the cache limit may be exceeded by single files.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubuntu

Syntax

```
utility <pathname>
```

Examples

This places the `RunQuiet.exe` program into the utility cache to avoid downloading it multiple times.

```
utility __Download/RunQuiet.exe
```

Prefetch a file, save the file to the utility cache as `patch.exe`, and run the command `patch.exe`.

```
prefetch patch.exe sha1:92c643875dda80022b3ce3f1ad580f62704b754f
 size:813160
http://example.com/foo.exe
utility __Download\patch.exe
wait __Download\patch.exe
```

Registry Commands

These commands allow you to edit the Windows Registry.

regdelete

Deletes a registry key value of the given name, regardless of whether it currently exists or not.

Version	Platforms
8.0.584.0	Windows

Syntax

```
regdelete <key> <value-name>
```

Where `key` is the name of the key and `value-name` is the name of the value in the registry key you wish to delete.

Examples

Deletes the `NeverShowExt` value from the `ShellScrap` registry key.

```
regdelete "[HKEY_CLASSES_ROOT\ShellScrap]" "NeverShowExt"
```

Notes:

This command is Windows-only. It will cause an action script to terminate on a Unix agent.

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: 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. You can use the appendfile or createfile until commands to build this .reg file. If the specified key doesn't already exist, it will be created by this command.

regdelete64

This command uses the same syntax as the regdelete command, but places a call to Wow64DisableWow64FsRedirection before launching the 64-bit version of regedit to edit the registry, allowing you to use the 64-bit registry available on 64-bit machines. This command 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.

Version	Platforms
8.0.584.0	Windows

Syntax

```
regdelete64 <key> <value-name>
```

Where `key` is the name of the key and `value-name` is the name of the value in the registry key you wish to delete.

Examples

Deletes the `NeverShowExt` value from the `ShellScrap` registry key.

```
regdelete64 "[HKEY_CLASSES_ROOT\ShellScrap]" "NeverShowExt"
```

Notes: This command is Windows-only. It will cause an action script to terminate on a Unix agent. If the specified key doesn't already exist, it will be created by this command.

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.

Version	Platforms
8.0.584.0	Windows

Syntax

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

Where `key` is the registry key of interest and `name` is the key value to set to `value`. These values are entered just as they are in a `REGEDIT4` registry file, in keeping with the rules for `regedit.exe`, the Windows program that edits the registry. String values are delimited by quotes, and the standard 4-byte integer `DWORD` is identified using dword: followed by the numeric value entered in hexadecimal with leading zeroes as shown below.

Examples

Set the `Level` value of the specified registry key to the `DWORD` 2.

```
regset "[HKCU\Software\Microsoft\Office\9.0\Word\Security]"
 "Level"=dword:00000002
```

Set the `testString` value of the specified registry key to `bob`.

```
regset "[HKEY_CURRENT_USER\Software\BigFix Inc.]" "testString"="bob"
```

Clear the data of the specified registry value.

```
regset "[HKEY_CLASSES_ROOT\ShellScrap]" "AlwaysShowExt"=""
```

Notes

This command is Windows-only. It will cause an action script to terminate on a Unix agent.

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 `REGEDIT4` registry files.

When you use the `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 back slashes `\` in the value field need to appear as double slashes `\\`.

For example, if you were trying to assign the value `SourcePath2` of the registry key `HKEY_LOCAL_MACHINE\Example` to `C:\I386`, the command that you would define would look like this:

```
regset "[HKEY_LOCAL_MACHINE\Example]" "SourcePath2"="C:\\I386"
```

Alternatively, you could use the escape relevance inspector:

```
regset "[HKEY_LOCAL_MACHINE\Example]" "SourcePath2"={escape of "c:\I386"}
```

In situations where you need to issue many regset commands, you might consider using the appendfile or createfile until commands to build a properly formatted regedit file, and then run regedit silently:

```
createfile until end-reg-edit-commands
REGEDIT4
[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion]
"SourcePath1"="c:\\I386"
"SourcePath2"="{escapes of pathname of windows folder}"
end-reg-edit-commands
move __createfile setup.reg
wait regedit /s setup.reg
```

If the specified key doesn't already exist, it will be created by this command.

regset64

This command uses the same syntax as the regset command, but places a call to Wow64DisableWow64FsRedirection before launching the 64-bit version of regedit.exe to

edit the registry. This allows you to use the native 64-bit registry to set a registry key to the given name and value. If the key doesn't already exist, this command creates the key with this initial value.

Version	Platforms
8.0.584.0	Windows

Syntax

```
regset64 <key> <name>=<value>
```

Examples

Set the `Level` value of the specified registry key to the `DWORD` 2:

```
regset64 "[HKCU\Software\Microsoft\Office\9.0\Word\Security]"
 "Level"=dword:00000002
```

Set the `testString` value of the specified registry key to `bob`.

```
regset64 "[HKEY_CURRENT_USER\Software\BigFix Inc.]" "testString"="bob"
```

Clear the data of the specified registry value.

```
regset64 "[HKEY_CLASSES_ROOT\ShellScrap]" "AlwaysShowExt"=""
```

Notes

This command is Windows-only. It will cause an action script to terminate on a Unix agent.

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 `REGEDIT4` registry files.

When you use the `regset64` 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 back slashes `\` in the value field need to appear as double slashes `\\`.

For example, if you were trying to assign the value `SourcePath2` of the registry key `HKEY_LOCAL_MACHINE\Example` to `C:\I386`, the command that you would define would look like this:

```
regset64 "[HKEY_LOCAL_MACHINE\Example]" "SourcePath2"="C:\\I386"
```

Alternatively, you could use the escape relevance inspector:

```
regset64 "[HKEY_LOCAL_MACHINE\Example]" "SourcePath2"={escape of "c:\I386"}
```

In situations where you need to issue many regset64 commands, you might consider using the appendfile or createfile until commands to build a properly formatted regedit file, and then run regedit silently:

```
createfile until end-reg-edit-commands
REGEDIT4
[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion]
"SourcePath1"="c:\\I386"
"SourcePath2"="{escapes of pathname of windows folder}"
end-reg-edit-commands
move __createfile setup.reg
wait regedit /s setup.reg
```

If the specified key doesn't already exist, it will be created by this command.

regkeydelete

Deletes a registry key and all of its contents.

Version	Platforms
9.5.13.130	Windows

Syntax

```
regkeydelete <key>
```

where `key` is the name of the registry key you wish to delete.

Examples

Deletes the `MyKey` registry key that is at [HKEY_LOCAL_MACHINE \SOFTWARE*WOW6432Node*].

```
regkeydelete "[HKEY_LOCAL_MACHINE\SOFTWARE\MyKey]"
```

Notes:

- This command is Windows-only. It will cause an action script to terminate on a Unix agent.
- Both your BigFix client and console should be at version 9.5.13 or later for the command to work.
- You cannot delete root keys (for example, HKEY_LOCAL_MACHINE).

regkeydelete64

This command uses the same syntax as the regkeydelete command, but places a call to Wow64DisableWow64FsRedirection before launching the 64-bit version of regedit to edit the registry, allowing you to use the 64-bit registry available on 64-bit machines. This command deletes a registry key and all of its contents. If the value doesn't already exist, this command fails and all subsequent commands are not run.

Version	Platforms
9.5.13.130	Windows

Syntax

```
regkeydelete64 <key>
```

Where `key` is the name of the key you wish to delete.

Examples

Deletes the `MyKey` registry key.

```
regkeydelete64 "[HKEY_LOCAL_MACHINE\SOFTWARE\MyKey]"
```

Notes:

- This command is Windows-only. It will cause an action script to terminate on a Unix agent.

- Both your BigFix client and console should be at version 9.5.13 or later for the command to work.
- If the specified key doesn't already exist, it will be created by this command.
- You cannot delete root keys (for example, HKEY_LOCAL_MACHINE).

Site Commands

These commands control site subscription and evaluation. They should not be used directly. Instead, use the console to manage site subscriptions.

site force evaluation

This command causes the client to re-evaluate all content in the site.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubunt

Syntax

```
site force evaluation
```

Examples

Force all content in the site to be re-evaluated.

```
site force evaluation
```

Notes: This command can place more load on the client machine, and should probably not be used.

subscribe

This command subscribes the client to the site identified in the masthead file.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows

Version	Platforms
8.1.535.0	Debian, Ubunt

Syntax

```
subscribe <masthead-file-name>
```

Examples

Subscribe the client computer to the site with the masthead `cool-site.fxm`.

```
subscribe "__Download\cool-site.fxm"
```

Notes: The command should not be used directly. Instead, use the BigFix Console to manage site subscriptions. This command returns an error unless it is executed in the master action site.

unsubscribe

This command unsubscribes from the current site. This command should not be used. Instead, manage site subscriptions from the BigFix Console.

Version	Platforms
8.0.584.0	AIX, HP-UX, Mac, Red Hat, SUSE, Solaris, Windows
8.1.535.0	Debian, Ubunt

Syntax

```
unsubscribe
```

Notes: The command should not be used directly. Instead, use the BigFix Console to manage site subscriptions.

Agent to Agent Communication

These commands can be used to pass instructions to a specific agent installed on the same endpoint as the BigFix agent. The Agent to Agent Communication channel (also

known as BigFix A2A) is implemented by DLL files put in place by the "Install BigFix A2A" Fixlet.

agent interface

This command allows to pass instructions to the other end of the Agent to Agent communication channel, addressing a specific agent.

VersionPlatforms

9.5.5.0 All

Syntax

```
agent interface "ProductID" command
```

Examples

These are examples using the ProductID "My_Prod":

```
agent interface "My_Prod" quarantine file -filepath "C:\myfolder
\myfile.exe"
agent interface "My_Prod" kill 10567
```

Chapter 4. Support

For more information about this product, see the following resources:

- BigFix Support Portal
- BigFix Developer
- BigFix playlist on YouTube
- BigFix Tech Advisors channel on YouTube
- BigFix Forum

Notices

This information was developed for products and services offered in the US.

HCL may not offer the products, services, or features discussed in this document in other countries. Consult your local HCL representative for information on the products and services currently available in your area. Any reference to an HCL product, program, or service is not intended to state or imply that only that HCL product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any HCL intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-HCL product, program, or service.

HCL may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

For license inquiries regarding double-byte character set (DBCS) information, contact the HCL Intellectual Property Department in your country or send inquiries, in writing, to:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

HCL TECHNOLOGIES LTD. PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. HCL may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-HCL websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this HCL product and use of those websites is at your own risk.

HCL may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by HCL under terms of the HCL Customer Agreement, HCL International Program License Agreement or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-HCL products was obtained from the suppliers of those products, their published announcements or other publicly available sources. HCL has not tested those products and cannot confirm the accuracy of performance, compatibility or

any other claims related to non-HCL products. Questions on the capabilities of non-HCL products should be addressed to the suppliers of those products.

Statements regarding HCL's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to HCL, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. HCL, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS," without warranty of any kind. HCL shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:
© (your company name) (year).
Portions of this code are derived from HCL Ltd. Sample Programs.

Trademarks

HCL Technologies Ltd. and HCL Technologies Ltd. logo, and hcl.com are trademarks or registered trademarks of HCL Technologies Ltd., registered in many jurisdictions worldwide.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Terms and conditions for product documentation

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

HCL reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by HCL, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

HCL MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.