Borland® StarTeam® 2009

StarTeam Command-line Tools Help

Borland®

Borland Software Corporation
8310 N Capital of Texas Hwy, Bldg 2, Ste 100
Austin, Texas 78731 USA
www.borland.com

June 2009
PDF

Command Line Tools

Command Line Tools

This section documents the various Command Line Operation Tools.

In This Section

Reference

This section contains all reference topics.

Reference

This section contains reference information.

In This Section

Command-line Operations
This section contains reference topics related to command-line operations.

Client Command-line Operations
This section contains reference topics related to client command-line operations.

VCM Command-line Utility
Explains the View Compare/Merge command-line utility (`VCMUtility`).

Command-line Operations

This section contains reference topics related to command-line operations.

In This Section

Bulk Check-out Utility Command-line Options
This topic describes the command-line options for the Bulk Check-out utility.

starteamserver Command Parameters
Describes and provides examples of the options used with the starteamserver command.

Check-out Trace Utility Command Line Operations
This topic describes the command-line options for the Check-out Trace utility.

Vault Verify Command-line Options
This topic describes the command-line options for the Vault Verify utility.

Bulk Check-out Utility Command-line Options

This topic contains the following information about BCO.

- Syntax
- Command-line Options
- BCO Usage Examples

Syntax

BCO uses the following syntax.

```
-p "projectSpecifier" [-pwdfile "filePath"] [-autoLogon] [-cwf] [-is] [-rp
"folderPath" | -fp "folderPath" ] [-cmp] [-dryrun] [-vb] [-useCA] [-encrypt
encryptionType] [-cfgl "labelName"| -cfgp "stateName" | -cfgd "asOfDate"] [-filter
"fileStatus"] [-o] [-ro] [-ts] [-fs] [-eol on | off | cr| lf| crlf ] [-exclude
<pattern> | @<pattern file>] [-netmon] [-t] [-h | -help] [files...]
```

Options

The table below describes options used with BCO.

Option	Description
-autoLogon	If a user name is not specified in the-p option, an attempt is made to logon using the user ID and password for the specified Server as stored by the Toolbar utility. This is available only on Windows operating systems.
-cfgd	Configures the view as of the specified date/time. Examples include: ■ "12/29/01 10:52 AM" ■ "December 29, 2001 10:52:00 AM PST" ■ "Monday, December 29, 2001 10:52:00 AM PST"
-cfgl	Configures the view using the specified label. Without -cfgl, -cfgp, or -cfgd, BCO uses the current configuration for the view.
-cfgp	Configures the view using the specified promotion state.
-cmp	Compresses all the data sent between the workstation and the server and decompresses it when it arrives. Without this option, compression does not take happen. Compression is most useful and appropriate when the client and server communicate over a slow connection. To determine whether to use compression, a small test case may be helpful. You must consider whether the time spent compressing and uncompressing data is better than the longer transfer time of uncompressed data sent over the slow connection.
-cwf	Create the working folder for the base folder and, if specifying the -is option, all subfolders of the base folder, even if they do not have any files to check-out during the run. Only visible folders are created.
-csf	When the command maps the folder specified in the -p option to the underlying StarTeam folder, using -csf causes the command to differentiate StarTeam folders based on the case-sensitive spelling of their names (This option does not apply to the case-sensitivity of filenames in the folders.) For example, with -

	`csf`, StarTeam folders named doc and Doc are recognized as different folders. Without this option, either folder could be recognized as the "doc" folder. The default is that StarTeam folders are not differentiated based on the case of letters in their names. With or without `-csf`, if folder names are ambiguous, an error occurs. For example, when you use `-csf`, the names of two folders are ambiguous if both a Doc and doc folder exist. When you do not use `-csf`, folder names are ambiguous if they are spelled identically.
`-dryrun`	Does not check out files, but displays a list of the files that would be checked out if `-dryrun` were not specified. The paths are those for the working folders into which the files would have been copied. When used with `-vb`, you get a complete picture of what would have happened.
`-encrypt RC4 | RC2_ECB | RC2_CBC | RC2_CFB`	Encrypts all the data sent between the workstation and the server and unencrypts it when it arrives. Without this option, encryption does not take place. Encryption protects files and other project information from being read by unauthorized parties over unsecured network lines. The full syntax is: `-encrypt encryptionType` BCO supports the following types of encryption: *RC4*: RSA RC4 stream cipher (fast) *RC2_ECB*: RSA RC2 block cipher (Electronic Codebook) *RC2_CBC*: RSA RC2 block cipher (Cipher Block Chaining) *RC2_CFB*: (Windows platforms only) RSA RC2 block cipher (Cipher Feedback) These encryption types are ordered from fastest to slowest. Each of the slower encryption types is safer than the one preceding it.
`-eol { on | off | cr | lf | crlf }`	Can automatically convert end-of-line markers. When specified with the `-on` option, text files are transferred from the Server's repository to the working folder on the workstation with the end-of-line convention for the platform executing the command as determined by the Java VM. When you do not specify this option, the default, BCO does not perform end-of-line conversion. Using the default is the same as not using `-eol` at all. When you specify the end-of-line character (`cr`, `lf`, or `crlf`), BCO transfers text files from the Server's repository to the working folder on the workstation with the specified end-of-line convention. For Windows platforms, the end-of-line marker is a carriage return/ line feed (`crlf`) combination; for UNIX platforms, it is a line feed (`lf`). You would set this option if, for example, when you compare a file from the repository and a working file on a UNIX system (if the repository stores text files as `crlf`).
`-exclude <pattern> | @<pattern file>`	Excludes files whose name or parent folder matches a given pattern. A pattern can be an exact file or folder name or it contain wildcard characters (e.g., '*.class'). To specify a folder name, precede the pattern name with a forward-slash (e.g., '/bin'). A single pattern can be provided with each `-exclude`, which can be repeated. Alternatively, one or more patterns can be specified on separate lines of the given `<pattern file>` (prefixed with `@`).
`files...`	Specifies the files to be used in the command by name or by filename-pattern specification (such as `"*.c"`). All options are interpreted using the semantic conventions of UNIX instead of Windows because conventions for UNIX are more specific. This means that `"*"`, rather than `"*.*"` means `"all files"`. The pattern `"*.*"` means `"all files with filename extensions"`. For

example, "star*.*" finds starteam.doc and starteam.cpp but not starteam. To find all of these, you could use "star*".

Without this option, the default is "*".

If you use * rather than "*" to indicate all files, a UNIX shell expands it into a series of items and passes this series as a group of options to the bco command. This can cause problems, for example, when you are checking out missing files, so it is best to use "*" to avoid unwanted complications.

If you use a set of file patterns, each pattern should be enclosed in its own set of quotation marks. For example, you can use "*.bat" "*.c", but you cannot use "*.bat *.c".

Note: Borland recommends that you enclose this option in quotation marks, regardless of platform, but for different reasons. On Windows platforms, file and folder names that contain spaces will not be interpreted correctly. On UNIX platforms, the shell will expand the option, then pass the list of items produced by the expansion to the client. Frequently, this produces unintended results. You can avoid both of these consequences by always enclosing this option in quotation marks. Only if it is essential that the option be expanded by the UNIX shell is it advisable to omit the quotation marks. Mysterious failures of batch scripts on either platform may be due to this omission.

Several special characters can be used in the file specification:

***** Matches any string including the empty string. For example, "*" matches any filename, with or without an extension. "xyz*" will match "xyz" and "xyz.cpp" and "xyzutyfj".

? Matches any single character. For example, "a?c" will match "abc" but NOT "ac".

[...] Matches any one of the characters enclosed by the left and right brackets.

A pair of characters separated by a hyphen (-) specifies a range of characters to be matched.

If the first character following the right bracket ([ ) is an exclamation point (!) or a caret (^), the rest of the characters are not matched. Any character not enclosed in the brackets is matched. For example, "x[a-d]y" matches "xby" but not "xey". "x[!a-d]y" matches "xey", but not "xby".

A hyphen (-) or right bracket ( ]) may be matched by including it as the first or last character in the bracketed set.

To use an asterisk (*), question mark (?), or left bracket ([ ) in a pattern you must precede it with the escape character, which is the backslash (\).

-filter	Specifies a string of one or more characters, each of which represents a file status. Never include spaces or other white space in this string. Only files that currently have the specified statuses will be checked out. You cannot check out files that are *Not In View*. The letters used to represent the statuses are: C for Current, M for Modified, G for Merge, O for Out of Date, I for Missing, and U for Unknown. If you use C, M, G, O, or U, you must also specify -o to force the check-out operation. Without the -o option specified, BCO skips files with the statuses represented by C, M, G, O, or U, but a warning is logged in the log file.
-fp "folderPath"	Overrides the working folder or working directory for the specified folder. This is equivalent to setting an alternate working path for the folder. While this enables you to use a different working folder than the one specified by the folder, its critical importance is its use to provide cross-platform compatibility. For example, UNIX and Windows systems specify drive and directory path names

	in incompatible ways. Although the path `"D:\MYPRODUCT\ DEVELOPMENT \SOURCE"` is understood on a Windows platform, it is not understood on a UNIX platform. Use this option to define the working path if your platform does not understand the path specified in the project. A backslash (`\`) is interpreted as an escape character when it precedes quotation marks. As a result, an error occurs in the following example: `bco -p "xxx" -fp "C:\" "*"` which is interpreted as: `bco -p "xxx" -fp "C:" *"`. To avoid a situation like this, escape the final character in `"C:\"` as follows: `bco -p "xxx" -fp "C:\\" "*"` . Or avoid it as follows when the `-fp` path does not end with the root folder as in `"C:\orion\"`: `bco -p "xxx" -fp "C:\orion" "*"` . Also note that UNC network paths require an additional escape character; for example: `bco -p "xxx" -fp "\\\server\path"`. The full syntax is: `-fp "folderPath"` . Folder is the Windows term and appears in the user interface. Directory is the correct term for the UNIX platform.
`-fs`	Prevents file statuses from being remembered after the check-out occurs. Subsequent status values for these files will be incorrect and indeterminate. Use this option where a file's status is irrelevant. For example, if you routinely delete the working folders before checking out files for a build, there are no files and their statuses do not matter. Be aware that the file statuses may never be known—even if you use the update-status command later. You can do a force check out without the `-fs` option to obtain current files with correct statuses.

`-h`
 	
`-help`
	Displays information about the command-line options.

`-is`	Applies the command recursively to all child folders. Without this option, the command applies only to the specified folder.
`-netmon`	Outputs SDK NetMonitor information to the console window. NetMonitor displays statistics for server commands. See the examples for BCO in the next section of this topic.
`-o`	Forces the check-out of files whose status would normally not allow them to be checked out. Those statuses are *Modified*, *Merge*, or *Unknown*.
`-p`	Indicates what view or folder is to be used, as well as providing the user name and password needed to access the server. The full syntax is: `-p "userName:password@hostName:portNumber/ projectName/ [viewName/][folderHierarchy/]"`. For example: `-p "JMarsh:password@orion:49201/StarDraw/ StarDraw/SourceCode/"`. If the user name is omitted, the current user name is used. The user name in the example is `"JMarsh"`.

	If the password is omitted, the user is prompted to enter the password. When the user types a password, the characters are not displayed on the screen. The password in the example is "password".
	If the host name is omitted, the default is localhost. The host name in the example is "orion".
	The port number is required. The default port number, 49201, is used in the example.
	The project name is always required. The project name in the example is StarDraw.
	Use a view hierarchy to identify the view. Use the colon (:) as a delimiter between view names. The view hierarchy should always include the root view. For example, "StarDraw:Release 4:Service Packs" indicates that the view to be used is the *Service Packs view*, which is a child of the *Release 4 view* and a grandchild of the *StarDraw* root view. If the view name is omitted, the root (default) view is used. If the view is the only view in that project with that name, you can use only the view name. (This is not recommended because another view with that name could be created at a later date and cause confusion.) The view name in the example is *StarDraw*. Because this is the root view of the *StarDraw* project, it could have been omitted.
	Use a folder hierarchy to identify the folder. Use the forward slash (/) as a delimiter between folder names. The folder hierarchy never includes the root folder. Omit the folder hierarchy if the file is in the view's root folder. For example, if the root folder of the view is *StarDraw* and the hierarchy to your files is "StarDraw/SourceCode/Client", use only "SourceCode/Client".
	If any of the parameters used with this option, for example, user names, view names, passwords, project names, or folder names contain characters that are used as delimiters, use the percent sign (%) followed by the hex code for each of those characters. For example, if "@" appears as a character in a password, you must replace it with "%40" .
	For ":", use "%3a".
	For "/", use "%2f" .
	For "@", use "%40" .
	For "%", use "%25".
	In UNIX and other operating systems, some special characters must be preceded by a backslash "\" or another escape character. In the -p option, you can replace such characters with hex codes. For example, "%3c" could be used in UNIX instead of "\<".
	For a space, use "%20".
	For "<", use "%3c".
	For ">", use "%3e".
-pwdfile	Specifies the path to a file that stores the password for the user. This option overrides the password used as part of the -p option. It prevents others from seeing the password for the user on the command line. It must be saved in UTF-8 format. The full syntax is: -pwdfile "filePath" .
-ro	Makes the working file read-only after this operation. Without this option, the file remains as it was prior to the operation. Usually, you use -ro to prevent yourself from editing a file that is not locked by you. Without -ro, the files are read/write.

-rp	Specifies or overrides the working folder or working directory for the root folder of the view. This is equivalent to setting an alternate working path for the view. Refer to the description of the -fp option in this table for additional information. The full syntax is: -rp "folderPath".
-t	Displays check-out volume and timing statistics.
-ts	Sets the time stamp for each working file to the check-out time. Without this option, the file is given the same time stamp as the checked-in revision of the file.
-useCA	Attempts to check out files using a StarTeamMPX Cache Agent. The full syntax is: -useCA host:port | autolocate. The host:port syntax specifies the host name (or IP address) and port number of the Cache Agent to be used. Alternatively, autolocate can be specified to automatically locate the nearest network Cache Agent. autolocate requires that the StarTeam Server is MPX-enabled.
-vb	Output is verbose. Displays each file as it is checked-out. The folder path is the folder path, rather than the working folder path.

BCO Usage Examples

The following example uses BCO to force check-out all the files from *Source Code*, a child of the root folder *StarDraw* (in the *StarDraw* view of the *StarDraw* project).

```
bco -p "JMarsh:password@Orion:1024/StarDraw/StarDraw/Source Code" -is -o "*"
```

The next example shows a BCO command that uses the -netmon option and the output displayed by Net Monitor.

```
bco -p "Administrator:Administrator@10.50.6.91:49201/StarDraw/WebSite" -fp D:\ Test
-netmon -o "*.htm"
```

Sample Output:

```
StarTeam BulkCheckOut Utility version 9.0.xxx
Copyright (c) 2006 Borland Software Corporation. All rights reserved.
Start: (rev 100) SRVR_CMD_GET_PROJECT_LIST Time: 62 millis; Sent: 42 bytes;
Got: 1834 bytes
Start: (rev 100) SRVR_CMD_GET_PROJECT_VIEWS Time: 47 millis; Sent: 46 bytes;
Got: 186 bytes
Start: (rev 100) SRVR_CMD_GET_PROJECT_VIEWS Time: 15 millis; Sent: 46 bytes;
Got: 186 bytes
Start: (rev 100) SRVR_CMD_PROJECT_OPEN Time: 188 millis; Sent: 70 bytes;
Got: 120 bytes
Start: (rev 100) PROJ_CMD_GET_VIEW_PROPERTIES Time: 31 millis; Sent: 42 bytes;
Got: 2556 bytes
```

```
Start: (rev 100) PROJ_CMD_GET_FOLDERS Time: 63 millis; Sent: 42 bytes;
Got: 1112 bytes
Start: (rev 100) PROJ_CMD_GET_FOLDER_ITEMS Time: 16 millis; Sent: 50 bytes;
Got: 40 bytes
Start: (rev 100) PROJ_CMD_REFRESH_ITEMS Time: 3562 millis; Sent: 122 bytes;
Got: 414 bytes
Start: (rev 100) SRVR_CMD_GET_PROJECT_VIEWS Time: 16 millis; Sent: 46 bytes;
Got: 186 bytes
Start: (rev 100) PROJ_CMD_GET_PROJECT_PROPERTIES Time: 31 millis; Sent: 42 bytes;
Got: 4797 bytes
Start: (rev 100) FILE_CMD_CHECKOUT Time: 47 millis; Sent: 78 bytes;
Got: 108 bytes
Start: (rev 100) FILE_CMD_CHECKOUT Time: 31 millis; Sent: 78 bytes;
Got: 1767 bytes
Start: (rev 100) FILE_CMD_CHECKOUT Time: 31 millis; Sent: 78 bytes;
Got: 1140 bytes
Start: (rev 100) SRVR_CMD_PROJECT_CLOSE Time: 15 millis; Sent: 62 bytes;
Got: 16 bytes
Start: (rev 100) SRVR_CMD_RELEASE_CLIENT Time: 31 millis; Sent: 42 bytes;
Got: 16 bytes
```

starteamserver Command Parameters

This section describes the options for the `starteamserver` command in alphabetical order, with examples of their uses.

-access Key

Use with: `-serial`.

See also: `-serial`, `-license`, and `-eval`.

Registers the Server as a licensed version. Use this option with the `-serial` option. The first time you start the Server, you must register the application as either a licensed version or an evaluation copy. If you need a serial number/access key combination or an evaluation key to extend your evaluation period, contact http://www.borland.com/us/company/how-to-buy.html .

Example:

```
starteamserver -serial 1234 -access 5678
```

-all

Use with: `-start`, `-stop`, and `-restart`.

Used in conjunction with the `-start` (or `-restart`) or `-stop` options. The `-start -all` options start all server configurations that have a status of Ready in the `starteam-server-configs.xml file`. The `-stop` and `-all` options stop all server configurations that have a status of Running.

Example:

```
starteamserver -stop -all
```

-attach "AttachmentsPath"

Use with: `-start` and `-restart`.

Specifies the attachments path for a server configuration.

The first time you start a server configuration, the system creates an Attachments child folder under the path you specify for RepositoryPath and stores the path to this folder in the database used by the server configuration. If you change the location of the Attachments folder, you can modify the attachments path in the database using the `-start` (or `-restart`), and `-attach` options from the command line.

You can also modify the attachments path on the General tab of the StarTeam Server Configuration dialog. The new attachments path will take effect the next time you start the server configuration.

Example:

```
starteamserver -start MyServer -attach "c:\My Server\Attachments"
```

-autorecover

Use with: `-start`.

See also: `-stoponerrors`.

The `-autorecover` option instructs the Server to attempt to make limited repairs where necessary during the verification process.

Example:

```
starteamserver -start MyServer -autorecover
```

-dsn "DataSourceName"

Use with: `-new`, `-edit`, `-start`, and `-restart`.

See also: `-t`, `-p,` , and `-u`.

Specifies the database connection information. Enter the existing ODBC data source name (DSN).

In releases 5.1 and 5.2, Oracle databases were accessed using the Oracle net service name that is stored in `$ORACLE_HOME/network/admin/tnsnames.ora`. This is no longer the case.

The value you specify for DBServerName is stored in the starteam-serverconfigs.xml file. You can review or modify the database connection information by using:

- The `-view` and `-edit` options from the command line.
- Database tab of the **StarTeam Server Configuration** dialog box in StarTeam Administration.
- Database tab of the <Server configuration> Properties dialog box in Server Administration.

Modifications take effect the next time you start the server configuration.

Example:

```
starteamserver -edit MyServer -dsn MyServerDSN
```

-edit "ConfigurationName"

Use with: `-name`, `-dsn`, `-u`, and `-p`.

Edits the session options for the specified server configuration. You can edit the following options: `-name, -dsn, -u, -p`. If the server configuration is running, you must shut it down before you can make any edits.

Example:

```
starteamserver -edit MyServer -name Portable -dsn RemoteServer
-u StarTeamAdmin
```

-eval Number

See also: `-serial`, `-access`, and `-license`.

Extends the evaluation period for an evaluation copy of the Server. The first time you start the Server, you must register the application as either a licensed version or an evaluation copy. If you need a serial number/access key

combination or an evaluation key to extend your evaluation period, contact http://www.borland.com/us/company/how-to-buy.html .

Example:

```
starteamserver -eval 01234567890
```

-help

Displays a message describing all of the command options.

Example:

```
starteamserver -help
```

-licenses

See also: `-serial`, `-access`, and `-eval`.

Displays license and registration information. If you are running a evaluation copy of the application, the system displays a message informing you of this. Otherwise, the system displays your serial number.

Example:

```
starteamserver -licenses
```

-list

Lists the server configurations defined in the starteam-server-configs.xml file and the status of each one. A server configuration can have one of the following statuses at any given point in time; Ready, Starting, Running, Disabled, and Stopping.

Example:

```
starteamserver -list
```

The Server displays a message similar to the following:

```
Configuration Status MyServer Ready StarDrawRepository Running
Portable Ready
```

-name "ConfigurationName"

Use with: `-edit`, `-start`, and `-restart`.

Renames a server configuration. This option is used in conjunction with the `-edit` option. The new server configuration name will take effect the next time you start the server configuration.

Example:

```
starteamserver -edit MyServer -name NewTeamServer
```

-new "ConfigurationName"

Creates a hive named DefaultHive for the new server configuration with the specified name and settings. This configuration uses a Native-II vault. This option produces the same result as selecting **New** on the Server Administration Tool menu, and using the wizard to create a new configuration.

A number of options can only be specified with `-new`. These are: `-c`, `-r`, and `-t`.

Example:

```
starteamserver -new NewServer1 -r "c:\new server\" -t 1 -dsn NewServerDSN
-u Admin -p password
```

-p "DBUserPassword"

Use with: `-new`, `-edit`, `-start`, and `-restart`.

See also: `-dsn`, `-p`, `-t`, and `-u`.

Specifies the password used to access the database. The value you specify for DBUserPassword is stored in the starteam-server-configs.xml file. Ensure that the password you specify is the correct one for the database user name. You can review or modify the password and user name using the `-view` and `-edit` options from the command line. Any modifications you make will take effect the next time you start the server configuration.

Example:

```
starteamserver -edit MyServer -u JodyK -p password
```

-q

Executes a command in quiet mode—that is, the system displays no screen output.

Example:

```
starteamserver -stop -all -q
```

-r "RepositoryPath"

Use with: `-new`.

Specifies the repository path for a new server configuration. If the repository path you specify does not exist, the system will create the appropriate folders the first time you start this server configuration.

The value you specify for `RepositoryPath` is stored in the `starteam-serverconfigs.xml` file. You can review the repository path using the `-view` option from the command line or in the application on the **General** tab of the StarTeam Server Configuration tool in the Server Administration Tool..

Warning: Do not use the Server home folder/directory as a server configuration repository path because the server configuration will not start.

Example:`starteamserver -new NewServer1 -r "c:\new server\" -t 1 -dsn NewServerDSN -u Admin -p password`

-remove "ConfigurationName"

Deletes the specified server configuration from the starteam-server-configs.xml file.

Example:

```
starteamserver -remove MyServer
```

-restart "ConfigurationName"

Stops and restarts the specified server configuration. Use this option after you make changes to a server configuration and want those changes to take effect. If the server configuration fails to restart, check the server log file for more information.

You can restart a server configuration and modify a number of its options at the same time. The following options can be used with the `-restart` option: `-all`, `-attach`, `-dsn`, `-name`, `-p`, `-tcpip`, and `-u`. You cannot use both the `-all` and the specific configuration name at the same time.

Example:

```
starteamserver -restart MyServer -tcpip StarTeamTCPIP -u SuperUser
-p SuperUserPassword
```

-serial Number

See also: `-access`, `-license`, and `-eval`.

Registers the Server as a licensed version. Use this option with the `-access` option. The first time you start the Server, you must register the application as either a licensed version or an evaluation copy. If you need a serial number/access key combination or an evaluation key to extend your evaluation period, contact http://www.borland.com/us/company/how-to-buy.html . The serial and access numbers in the example below would be replaced with actual serial and access numbers.

Example:

```
starteamserver -serial 1234567890 -access 9999999
```

-start "ConfigurationName"

See also: `-all` and `-stop`.

Starts the specified server configuration. starteamserver updates the server configuration entry in the `starteam-server-configs.xml` file to `Status=Running` and `PID=nnn` where `nnn` would be replaced with the actual `PID` number.

You can start a server configuration and modify a number of its options at the same time. The following options can be used with the `-start` option: `-attach`, `-dsn`, `-name`, `-p`, `-tcpip`, and `-u`.

Example:

```
starteamserver -start MyServer -tcpip StarTeamTCPIP -u SuperUser
-p SuperUserPassword
```

-stop "ConfigurationName"

See also: `-all` and `-start`.

Shuts down the specified server configuration. After the server configuration stops running, starteamserver updates the entry in the `starteam-server-configs.xml` file to **Status=Ready** and `PID=0`.

Example:

```
starteamserver -stop MyServer
```

Note: For enterprise advantage users: If you are running the Server as a service and Notification Agent as a dependent service, you cannot shut down the Team Server unless the Notification Agent service is shut down first.

-t DBType

Use with: `-new`

See also: `-dsn`, `-p`, `-u`.

Specifies the database type. This option can be used only when you are creating a new server configuration. Use one of the following numbered values to indicate the type of database:

2 = Microsoft SQL Server or SSE

3 = Oracle

The value you specify for DBType is stored in the `starteam-server-configs.xml` file. You can review the database type using:

- The `-view` option from the command line.
- In StarTeam Administration Tool on the **Database** tab of the **StarTeam Server Configuration** tab.
- In Server Administration on the **Database** tab of the `<server configuration="">` **Properties** dialog box

Example:

```
starteamserver -new NewServer1 -r "c:\new server\" -t 1 -dsn NewServerDSN
-u Admin -p password
```

-tcpip Endpoint | up[:Endpoint] | down[:Endpoint]

Use with: `-start, -restart`

Sets the endpoint for the TCP/IP (Sockets) protocol. Also enables or disables the protocol. Use up to enable and down to disable. You can both set the endpoint and enable or disable it using up or down followed by a colon and the endpoint.

The value you specify for the endpoint is stored in the database used by this server configuration.

You can modify this information using the `-start` (or `-restart`) and `-tcpip` options from the command line or in the application on the **Protocol** tab of the **StarTeam Server Configuration** tab.

Example:

```
starteamserver -start MyServer -tcpip 49201 starteamserver -start
MyServer -tcpip up
```

-u "DBUserName"

Use with: `-new`, `-edit`, `-start`, `-P`, and `-restart`.

See also: `-t`, `-dsn`, and `-p`.

Specifies the user name that the server configuration uses to access the database. The value you specify for `DBUserName` is stored in the `starteam-server-configs.xml` file. You can review or modify the database user name using the `-view` or `-edit` options from the command line. Be sure to also specify the password for this user account. Any modifications you make will take effect the next time you start the server configuration. Ensure that the user name and password you specify using the `starteamserver` command is a valid account in the database. The server configuration will fail to start if the user account is missing in the database.

Example:

```
starteamserver -edit MyServer -u SuperUser -p SuperUserPassword
```

-version

Displays the version and build number for the Server.

Example:

```
starteamserver -version
```

The Server displays a message similar to the following:

```
StarTeam Server Version: x.x Build number: x.x.xxx
```

-view "ConfigurationName"

Lists the session properties of the specified server configuration.

Example;

```
starteamserver -view StarDraw
```

Related Reference

Command-line Operations

Check-out Trace Utility Command Line Operations

This topic describes the command-line options for the Check-out Trace utility.

In general, you can run the utility from the command line with default options as follows: `CheckoutTraceDump.exe -go` . Valid options for Check-out Trace are described in the following table.

Option	Description
`-go`	Specify this flag to run with default settings.
`-path:<path>`	Folder of the binary check-out trace (`.cotrc`) files. Defaults to the current folder.
`-outpath:<path>`	Folder for the output (`.csv`) files. Defaults to the same folder as the binary trace (`.cotrc`) files.
`-file:<filespec>`	Binary check-out trace files to be used as input. Supports standard file system wildcard values (*,?). Defaults to `"*.cotrc"` (all check-out trace files in the folder). You cannot use more than one path with the parameter, and you cannot specify this parameter more than once per command.
`-ext:<extension>`	The file extension used for check-out trace files. The extension is appended to the binary check-out trace (`.cotrc`) filename to create the output dump filename. Defaults to `.csv`.
`-start:<start time>`	Earliest date-time of interest. Only checkouts that occurred after this time will be output. By default, the utility does not filter by time.
`-end:<end time>`	Specifies the most recent date-time of interest. Only checkouts that occurred before this time will be output. By default, the utility does not filter by time.
`-project:<project name>`	Name of the project for which check-out information is to be filtered. Only checkouts from this project will be output. By default, the utility does not filter by project. All projects are included in the output. If both `-project` and `-projectid` are specified, `-projectid` takes precedence.
`-projectid:<project ID>`	ID of the project for which check-out information is to be filtered. Only checkouts from this project will be output. By default, the utility does not filter by project ID. All projects are included in the output. This option takes precedence over `-project` if both options are specified.
`-separator:<separator>`	String used to separate values in the output file. By default, the utility uses `","`.
`-overwrite`	Specify this flag to overwrite existing check-out trace files. If not specified, check-out trace binary files will be skipped if a trace dump file with the target name already exists.

Vault Verify Command-line Options

This topic describes the command-line options for the Vault Verify utility.

In general, you can run Vault Verify from the command line as follows: `VaultVerify [options] "configuration"`.

Based on the default or given `-check` options, integrity checks are performed on the vault archive files for the specified StarTeam "server configuration". If you specify the `-repair` option, Vault Verify attempts to correct problems found. Vault Verify opens the database for the server configuration but does not modify it. Valid options for Vault Verify are described in the following table.

Option	Description
`-check {missing | corrupt | stray | all}`	Determines which integrity checks to perform: `missing`: Checks for missing files by comparing the database against archive files actually present. `corrupt`: Checks the integrity of existing archive files (MD5, name, folder, and .gz file format). `stray`: Checks for extraneous files based on the database. This option cannot be used if the server configuration is in use. `all`: Performs all integrity checks. Multiple `-check` options can be specified. Also, see the `-repair` option.
`-cf <folder path>`	Path name of the *corrupt file folder*, where problem files found by the `corrupt` check are moved when `-repair` is specified. The default corrupt file folder is `C:\Temp\VVCorruptFiles`.
`-dbhost <host>`	Specifies the host name of the database for the specified <server configuration>. On Windows, it is only meaningful when `-dbinstance` is also provided. On Windows and Linux, use this option only when the database server executes on a different host than this one.
`-dbname <name>`	Specifies the database name for the specified <configuration>. On Windows, this parameter is only meaningful when `-dbinstance` is also specified, and it is only needed when the database name is different than the ODBC DSN. On Linux, use this option only if `-dbinstance` is not used and the Oracle service name is different than the TNS name.
`-dbinstance <name>`	This option is only meaningful on Windows. When used, it causes VaultVerify to open the database directly instead of via the ODBC DSN specified in the configuration file. For SQL Server, the <name> must be the instance name (e.g., 'SSE2005_ST'). Note that the default Instance name for SQL Server is 'MSSQLSERVER' and for SQL Server Express, it is 'SQLEXPRESS'. For Oracle, should be the service name, (e.g., 'ORCL'). `-dbinstance` must be used with `-dbhost` when the database server executes on a different host. For SQL Server, `-dbname` should also be used if the database name is different than the ODBC DSN. For Oracle, `-dbname` is ignored if `-dbinstance` is specified.
`-dbpassword <password>`	Specifies the database logon password. If not specified, a blank password is used. (The password stored in the configuration is encrypted and cannot be used by Vault Verify.) On server

	configurations running against Oracle, this option must be specified since the Oracle password is never empty.
`-dbport <port>`	Specifies the TCP/IP port to use to connect to the database server. This parameter is only used on non-Windows platforms when a different port is used than the vendor's default database port (e.g., 1521 for Oracle).
`-dbuser <user>`	Specifies the logon ID used to connect to the database. If specified, this parameter overrides the user specified in the StarTeam <configuration>. The only valid user to use with this option is the user that owns the StarTeam tables.
`-help (or -h or -?)`	Displays this usage information.
`-path <folder path>`	Specifies the folder path of the `starteam-server-configs.xml` file. This file must exist and contain the specified <server configuration>. By default, this file is opened in the parent folder of the current working directory if it is not found in the current working directory.
`-nosharereport`	Suppresses the reporting of share information. Normally, all share paths of each corrupt file is reported. This option suppresses the share path information, which can speed up application execution and substantially reduce the report size.
`-repair`	Specifies that an attempt should be made to correct archive file problems. 'Corrupt' archives are moved to the 'corrupt file folder' (see the `-cf` option). If they correspond to valid file revisions, they are then treated as missing. Missing archive recovery is attempted from other vault files and, if the `-useca` option is specified, from a Cache Agent. Stray archives are moved to the 'stray file folder' (see the `-sf` option). **Note:** `-repair` is ignored if the StarTeam <configuration> is in use.
`-sf <folder path>`	Path name of the 'stray file folder', where extraneous files found by the 'stray' check are moved when `-repair` is specified. The default 'stray file folder' is `C:\Temp\VVStrayFiles`.
`-t`	Displays elapsed time information when the verification finishes.
`-useca <host>:<port>`	If `-repair` is specified, this option enables attempts to recover missing files from the specified MPX Cache Agent. The `<host>` and `<port>` must designate a remote Cache Agent because it maintains an independent cache.
`-verbose`	Displays additional status information as the verification proceeds.
`"configuration"`	Specifies the configuration name. The configuration name passed to Vault Verify is case-sensitive, and if it includes spaces, you must pass the configuration name to Vault Verify in quotation marks.

Client Command-line Operations

This section contains reference topics related to client command-line operations.

In This Section

Common Options
Describes the options that show up in all or almost all commands.

Special Characters
Describes the special characters that you can use to search for files.

Exit Codes
Describes the exit codes returned by the stcmd commands.

Add Files: stcmd add
Describes the command to add files to a project.

Add Folders: stcmd add-folder
Describes the command to add StarTeam folders to a view.

Add Projects: stcmd add-project
Describes the command to add a project to a StarTeam Server configuration.

Add Views: stcmd add-view
Describes the command to add a view to a StarTeam server configuration.

Apply Labels: stcmd apply-label
Describes the command used to label specified file revisions with view or revision labels.

Check In Files: stcmd ci
Describes the command to check files into a StarTeam repository (or vault) from a working folder.

Check Out Files: stcmd co
Describes the command to check out files from a StarTeam repository (or vault) to your working folder.

Compare File Revisions: stcmd diff
Describes the command to display differences between two revisions of a file.

Change File Descriptions: stcmd dsc
Describes the command to change a file description.

Create Labels: stcmd label
Describes the command to create a view or revision label.

Create Working Folders: stcmd local-mkdir
Describes the command to create the working folder or working directory on your workstation for the specified StarTeam folder.

Delete Local Files: stcmd delete-local
Describes the command to delete files from a working folder.

Display File History: stcmd hist
Describes the command to display the revision history of files.

List Files: stcmd list
Describes the command to list all files in the folder specified by the -p option.

Lock and Unlock Files: stcmd lck
Describes the command to lock or unlock files.

Lock and Unlock Server: stcmd server-mode
Describes the command to lock and unlock a server configuration.

Remove Files: stcmd remove
Describes the command to remove files from version control.

Set Personal Options: stcmd set-personal-options
Describes the command to set and list personal options.

Update File Status: stcmd update-status
Describes the command to update the status of a file.

Common Options

Some options show up in all or almost all of these commands. In each command, they have exactly the same meaning, so they are explained in this section and not repeated later. Options that do not appear in all commands or vary in meaning from command to command are explained with the commands.

All command-line syntax is interpreted as UNIX rather than Windows syntax.

All options can be written with either a hyphen (`-`) or slash mark (`/`), depending on what your operating system understands. When this chapter indicates that an option requires quotations marks, using them is recommended, despite the fact that the option may be accepted by your operating system without them. Adopting this policy is both consistent and safe.

Windows operating systems require quotation marks when a space is part of the option. For example, when a revision comment is several words, it must be enclosed in quotation marks. A one-word comment does not require quotation marks, although quotation marks can be used. Be aware that commands sent via the Java command line fail when there are spaces in the names of working folders and files. You cannot use spaces in UNIX names unless they are enclosed in double quotation marks.

Syntax conventions

The syntax conventions for the command line are as follows:

[ ]

Square brackets identify optional syntax.

|

A vertical bar separates mutually exclusive choices. Select only one of the choices.

-?

Lists the command's syntax and a description of each option.

`-?` also invokes help. `-help` and `-h` are synonyms for`-?`. `-?` works with each command, although not shown in the syntax. This information is sent to `stderr` rather than `stdout`. To capture `stderr` information from the Windows command prompt, use `"2>"`, rather than `">"` which captures `stdout` information.

-active

Indicates the active process item.

-cmp

Compresses all the data sent between the workstation and the server and decompresses it when it arrives. Without this option, no compression takes place.

Compression is most useful and appropriate when the client and server communicate over a slow connection. To determine whether to use compression, a small test case may be helpful. You must consider whether the time spent compressing and uncompressing data is better than the longer transfer time of uncompressed data sent over the slow connection.

-csf

When the command maps the folder specified in the `-p` option to the underlying StarTeam folder, using `-csf` causes the command to differentiate StarTeam folders based on the case-sensitive spelling of their names This option does

not apply to the case-sensitivity of filenames in the folders. For example, with `-csf`, StarTeam folders named “doc” and “Doc” are recognized as different folders. Without this option, either folder could be recognized as the "doc" folder.

The default is that StarTeam folders are not differentiated based on the case of letters in their names.

With or without `-csf`, if folder names are ambiguous, an error occurs. For example, when you use `-csf`, the names of two folders are ambiguous if both a Doc and doc folder exist. When you do not use `-csf`, folder names are ambiguous if they are spelled identically.

-encrypt

Encrypts all data sent between the workstation and the server and decrypts it when it arrives. Without this option, no encryption takes place. Encryption protects files and other project information from being read by unauthorized parties over unsecured network lines.

The full syntax is:

```
-encrypt encryptionType
```

The types of encryption are:

RC4: RSA RC4 stream cipher (fast)

RC2_ECB: RSA RC2 block cipher (Electronic Codebook)

RC2_CBC: RSA RC2 block cipher (Cipher Block Chaining)

RC2_CFB: (Windows platforms only) RSA RC2 block cipher (Cipher Feedback)

These encryption types are ordered from fastest to slowest. Each of the slower encryption types is safer than the one preceding it.

Note: For platforms other than Windows, the public and private keys used in the encryption process are not created automatically. They are stored in a file in the user’s home directory. This options file is named `.starteam`. It contains a variable or shell variable called `keyfile`. The `keyfile` variable specifies the location of the file that contains the public and private keys. If you do not specify the `keyfile` variable, an error occurs. When you specify the `keyfile` variable, but the file does not exist, the StarTeam Client generates a random pair of keys, creates the file, and stores the keys in it. Be sure to secure this file. For example, in UNIX, only its owner should be able to read it.

-eol [on|off]

Perform end-of-line conversion of text files.

-epwdfile

Store's a user's password as an encrypted value in a local file. This feature supports automated build scripts, which must run unattended. The script would call `stcmd` with a specific user name and the filename containing the encrypted password for that user name. The password is then internally decrypted and passed to the server without being transmitted across the network as clear text.

The `-epwdfile` keyword specifies the path to the file that contains the encrypted password. Like `-pwdfile`, `-epwdfile` overrides the password being used as part of the -p otion, preventing others from seeing the user's password on the command line. The full syntax is:

```
-epwdfile "filePath"
```

The following table gives the syntax of the commands that can be used to store an encrypted password.

Action	Syntax
To be prompted for the password that will be encrypted and stored in a file.	`stcmd store-password -epwdfile "filePath"`
To include the encrypted password in the command as clear text. (Note that this action does not access the network with the clear value.)	`stcmd store-password -epwdfile "filePath" -password "password"`

Once an encrypted password is stored, other `stcmd` commands can specify `-epwdfile` "filePath"' as parameters. For example:

```
stcmd delete-local -p "JMarsh:password@Orion:1024/StarDraw/StarDraw/SourceCode" -epwdfile
"C:\estuff\myfile.txt" -filter "N" "*"
```

-f NCI

Apply command to all files needing check in.

`-f NCI` is ignored if `-filter` is used.

files...

Specifies the files to be used in the command by name or by file name-pattern specification, such as `"*.c"`. All options are interpreted using the semantic conventions of UNIX instead of Windows because UNIX conventions are more specific. This means that `"*"`, rather than `"*.*"` means “all files.” The pattern `"*.*"` means “all files with file name extensions.” For example, `"star*.*"` finds `starteam.doc` and `starteam.cpp`, but not `starteam`. To find all of these, you could use `"star*"`.

Without this option, the default is `"*"`. When used, this option must always be the last option. Any options after it are ignored.

If you use `*`, rather than `"*"` to indicate all files, a UNIX shell expands it into a series of items and passes this series as a group of options to the `stcmd` command. This can cause problems, for example, when you are checking out missing files, so it is best to use `"*"` to avoid unwanted complications.

If you use a set of file patterns, each pattern should be enclosed in its own set of quotation marks. For example, you can use `"*.bat" "*.c"`, but you cannot use `"*.bat *.c"`.

Note: We recommend that you enclose this option in quotation marks, regardless of platform, but for different reasons. On Windows platforms, file and folder names that contain spaces will not be interpreted correctly unless you use quotation marks. On UNIX platforms, if you do not use quotation marks, the shell will expand the option, then pass the list of items produced by the expansion to the client. Frequently this produces unintended results. You can avoid both of these consequences by enclosing the option in quotation marks. Only if it is essential that the option be expanded by the UNIX shell is it advisable to omit the quotation marks. Mysterious failures of batch scripts on either platform may be due to this omission.

Several special characters can be used in the file specification:

`*`

Matches any string including the empty string. For example, `*` matches any file name, with or without an extension. "xyz*" will match "xyz" and "xyz.cpp" and "xyzutyfj".

`?`

Matches any single character. For example, "a?c" will match "abc" but NOT "ac"

`[...]`

Matches any one of the characters enclosed by the left and right brackets.

A pair of characters separated by a hyphen (`-`) specifies a range of characters to be matched.

If the first character following the right bracket (`[`) is an exclamation point (`!`) or a caret (`^`), the rest of the characters are not matched. Any character not enclosed in the brackets is matched. For example, "x[a-d]y" matches "xby" but not "xey". "x[!a-d]y" matches "xey" but not "xby".

A hyphen (`-`) or right bracket (`]`) may be matched by including it as the first or last character in the bracketed set.

To use an asterisk (`*`), question mark (`?`), or left bracket (`[`) in a pattern, you must precede it with the escape character (which is the backslash (`\`).

-filter

File status filter. Statuses are `C` = Current, `M` = Modified, `O` = Out of Date, `N` = Not in View, `I` = Missing, `G` = Merge, and `U` = Unknown. For example, using `CM` applies a command only to files with a status of Current or Modified.

`-filter` takes precedence over `-f NCI`. If you use `G`, `O`, or `U`, you must also specify `-i` or `-o`. Otherwise the `G`, `O`, or `U` is ignored.

`-filter` also takes precedence over `-f NCO`. If you use `G`, `M`, `O`, or `U`, you must also specify `-merge` or `-o` to force the checkout operation. Otherwise, the `G`, `M`, `O`, or `U` is ignored.

-fp

Overrides the specified StarTeam folder's working folder or working directory. This is equivalent to setting an alternate working path for the folder.

While this option allows you to use a different working folder than the one specified by the StarTeam view, its critical importance is to provide cross-platform compatibility. For example, UNIX and Windows systems specify drive and directory path names in incompatible ways.

While the path `D:\MYPRODUCT\DEVELOPMENT\SOURCE` is understood on a Windows platform, it is not understood on a UNIX platform. Use this option to define the working path if your platform does not understand the path specified in the StarTeam project.

A backslash (`\`) is interpreted as an escape character when it precedes quotation marks. As a result, an error occurs in the following example:

```
stcmd ci -p "xxx" -fp "C:\" "*"
```

which is interpreted as:

```
stcmd ci -p "xxx" -fp "C:" *"
```

To avoid a situation like this, escape the final character in "C:\" as follows:

```
stcmd ci -p "xxx" -fp "C:\\" "*"
```

Or avoid it as follows when the `-rp` path doesn't end with the root folder as in `C:\orion\`:

```
stcmd ci -p "xxx" -fp "C:\orion" "*"
```

The full syntax is:

```
-rp "folderName"
```

Folder is the Windows term and appears in the StarTeam user interface. Directory is the correct term for the UNIX platform.

-h

Invokes help. `-help` works with each command, although not shown in the syntax.

-help

Invokes help. `-help` works with each command, although not shown in the syntax.

-i

Prompts user to confirm check-in when file status is Merge, Out of Date, or Unknown.

-is

Applies the command to all child folders. Without this option, the command applies only to the specified folder.

When this option is used with add-folder, you can add an entire branch of folders to the StarTeam folder hierarchy. When it is used with add-project, you can create a project with more than just a root folder.

-l

Locks a file.

-mark

Marks a change request as fixed, a requirement as complete, or a task as finished.

-nel

Non-exclusively locks a file

-nologo

Supresses the copyright notice. `-nologo` works with each command, although not shown in the syntax.

-nomove

Do not move labels if already attached.

-o

Forces check-in.

-p

Indicates the view or folder to be used; also provides the user name and password needed to access the server. The full syntax is:

```
-p "userName:password@hostName:endpoint/projectName/[viewName/][folderHierarchy/]"
```

For example:

```
-p “bsmith:rocketfive@orion:49201/StarDraw/StarDraw/SourceCode/”
```

- If the user name is omitted, the current user name is used.
- If the password is omitted, the user is prompted to enter the password. When the user types a password, the characters are not displayed on the screen.
- If the host name is omitted, the default is localhost.
- Entering an endpoint (port number) is required. The default is 1024.
- The project name is always required.
- A view hierarchy should be used to identify the view. Use the colon (`:`) as a delimiter between view names. The view hierarchy should always include the root view. For example, "StarDraw:Release 4:Service Packs" indicates that the view to be used is the Service Packs view, which is a child of the Release 4 view and a grandchild of the StarDraw root view. If the view name is omitted, the root view is used. If the view is the only view in that project with that name, you can use only the view name. Doing this is not recommended, however, because another view with that name could be created at a later date.
- A folder hierarchy should be used to identify the folder. Use the forward slash (`/`) as a delimiter between folder names. The folder hierarchy never includes the root folder. Omit the folder hierarchy if the file is in the view’s root folder. For example, if the root folder of the view is `StarDraw`, and the hierarchy to your files is `StarDraw/SourceCode/Client`, use only "SourceCode/Client".

If any of the variables used with this option contain characters that are used as delimiters, use the percent sign (`%`) followed by the hex code for each of those characters. For example, if “`@`” appears as a character in a password, you must replace it with "`%40`".

For "`:`", use "`%3a`"

For "`/`", use "`%2f`"

For "`@`", use "`%40`"

For "`%`", use "`%25`”

In UNIX and other operating systems, some special characters must be preceded by a backslash "`\`" or another escape character. In the `-p` option, you can replace such characters with hex codes. For example, "`%3c`" could be used in UNIX instead of "`\<`".

For a space, use "%20"

For "<", use "%3c"

For ">", use "%3e"

-pwdfile

Specifies the path to a file that stores the user's password. This option overrides the password used as part of the -p option. It prevents others from seeing the user's password on the command line. The full syntax is:

```
-pwdfile "filePath"
```

-q

Suppresses progress reporting. Without this option, messages about each action appear on the screen as the action is performed.

-r

Reason for check-in.

-rf

Precedes name of file that contains the reason for the check-in.

-ro

Sets file as read-only after operation.

-rp

Specifies or overrides the working folder or working directory for the StarTeam view's root folder. The stcmd add-project command uses this option to specify the working folder for the new view's root folder. Other commands use it to override the existing working folder.

While this option allows you to use a different working folder than the one specified by the StarTeam view, its critical importance is to provide cross-platform compatibility. For example, UNIX and Windows systems specify drive and directory path names in incompatible ways.

While the path D:\MYPRODUCT\DEVELOPMENT\SOURCE is understood on a Windows platform, it is not understood on a UNIX platform. Use this option to define the working path if your platform does not understand the path specified in the StarTeam project.

The UNIX shell interprets a backslash (\) as an escape character when it precedes certain characters, such as quotation marks. As a result, an error occurs in the following example:

```
stcmd ci -p "xxx" -rp "C:\" "*"
```

which is interpreted as:

```
stcmd ci -p "xxx" -rp "C:" *"
```

To avoid a situation like this, escape the final character in "`C:\`" as follows:

```
stcmd ci -p "xxx" -rp "C:\\" "*"
```

Or avoid it as follows when the `-rp` path doesn't end with the root folder as in "`C:\orion\`":

```
stcmd ci -p "xxx" -rp "C:\orion" "*"
```

The full syntax is:

```
-rp "folderName"
```

Folder is the Windows term and appears in the StarTeam user interface. Directory is the correct term for the UNIX platform.

-rw

Sets file as read-write after operation.

-stop

Often used with `-x`. Halts execution of the command-line when the first error is encountered. Without this option, execution continues despite errors.

-u

Unlocks a file.

-v

Version label.

-x

Switches between interactive and batch modes. Without this option, you must confirm error messages interactively, and the exit codes may not be available. With this option, no error messages are displayed, but exit codes are set. The exit codes are `0` for success and `1` for failure.

Related Concepts

Special Characters

Related Reference

Client Command-line Operations

Special Characters

* Matches any string including an empty string. For example, "x*z" will match "xyz" and "xz". ? Matches any single character. For example, "a?c" will match "abc" but NOT "ac".

[...] Matches any one of the characters enclosed by the left and right brackets.

A pair of characters separated by a hyphen(-)specifies a range of characters to be matched. If the first character following a left bracket ([ ) is an exclamation point (!) or a caret (^), the rest of the characters are not matched. Any character not enclosed in the brackets is matched. A hyphen (-) or right bracket ( ]) may be matched by including it as the first or last character in a bracketed set. For example, "x[a - d]y" matches "xby" but not "xey" while "x[!a - d]y" matches "xey" but not "xby". If you want to use an asterisk (*), question mark (?), or left bracket ([ ) in a pattern, you must precede it with the escape character (that is, a backslash \ ).

If you use * rather than “*” to indicate all files, a UNIX shell expands it into a series of items and passes this series as a group of options to the `stcmd` command. This can cause problems (for example, when you are checking out missing files) so it is best to use "*" and avoid unwanted complications. If you use a set of file patterns, each pattern should be enclosed in its own set of quotation marks. For example, you can use "*.bat" "*.c", but you cannot use "*.bat *.c".

These special characters also apply to the `files...` option available in some commands.

Related Reference

Client Command-line Operations

Exit Codes

The `stcmd` commands return exit codes if the `-x` option is used in the command. The codes are: 0 for success, 1 for failure, 101 if at least one of the specified file patterns did not match, 102 if none of the specified file patterns matched. The `stcmd diff` command has an additional option (`-e`) that returns exit codes. The `-e` option has three exit codes (0, 1, and 2) with meanings that are different from those listed above. In addition, the 1 might not be returned if you don't also use the `-x` option.

For Windows platforms

You can use `ERRORLEVEL` in a batch file to perform operations based on the result of a command. For example, after an `stcmd` command in a batch file, you might use the following:

```
IF ERRORLEVEL int statement
```

where int is 0 or 1.

For example:

```
IF NOT ERRORLEVEL 1 GOTO OPOK
ECHO ERROR OCCURRED AT STEP5>LOGFILE.TXT.

:OPOK
```

You can also use the pseudo environment variable `%ERRORLEVEL%`. For example, you might use the following in a shell or at the command line (after an `stcmd` command):

```
SET /A STEPNUMBER=5
SET /A THISERROR=STEPNUNBER*ERRORLEVEL
SET /A ERRORMASK=+THISERROR
```

For UNIX

Each shell has its own method of evaluating exit codes. For example, in the Bourne shell, the following statement might come after an `stcmd` command:

```
if [ return ]; then statement
```

Related Reference

Client Command-line Operations

Add Files: stcmd add

Use `stcmd add` to add files to a project from the command line.

You can simultaneously link the added files to a process item. All the files successfully added using this command will be linked and pinned to the tip revision of the process item. Use the `-active` option to specify the currently active process item (previously set using a StarTeam client on your workstation).

If no item is active or you prefer to use another item, use the option that indicates the type of the process item (`-cr`, `-req`, or `-task`), followed by the complete path from the root folder of the StarTeam project view to the item, using the forward slash (/) as a delimiter between folder names. For out-of-view process items, specify the project name and view name in front of the complete folder path. Separate the view path with a colon (:). For example, `-cr MyProject/RootView:ChildView/SourceCode/37` specifies change request 37 in the `SourceCode` folder of the `ChildView` view in the `MyProject` project. During execution, the process first assumes that the process item is in the current view, and it checks the current view to determine whether the full path corresponds to a folder path within that view. If the process item is not found in the current view, it is treated as an out-of-process item, and the search for the process item begins from the project and view.

Use the `-mark` option to simultaneously mark the process item as fixed, finished, or complete, depending on its type. For example, a change request can be marked as fixed. The item is not marked as fixed, finished, or complete unless all the files are successfully added.

Syntax

The syntax for this command is as follows:

```
stcmd add -p "projectSpecifier" [-pwdfile "filePath"] [-cmp] [-csf] [-encrypt
encryptionType]
[-is] [-q] [-x] [-stop] [-rp "folderPath" | -fp "folderPath"] [- l | -u | -nel] [-ro | -rw]]
[-d "description"]
[-vl "labelName"] [-eol [on | off]] [[ -active | [-cr | -req | -task ] processItemPath] [-
mark]] [-short] [files...]
```

Option	Description
`-active`	The active process item.

`-cr`
`-req`
`-task`
	Complete path from the project view's root folder to the change request, requirement, or task number to be used as a process item. Use the forward slash (/) as a delimiter between folder names.
	For out-of-view process items, specify the project name and view name in front of the complete folder path. Separate the view path with a colon (:). For example, `-cr MyProject/RootView:ChildView/SourceCode/37` specifies change request 37 in the `SourceCode` folder of the `ChildView` view in the `MyProject` project.
	During execution, the process first assumes that the process item is in the current view, and it checks the current view to determine whether the full path corresponds to a folder path within that view. If the process item is not found in the current view, it is treated as an out-of-process item, and the search for the process item begins from the project and view.

`-d`	Sets the description of the newly added files to the specified description. The description is enclosed in double quotation marks.
`-eol`	Automatically converts end-of-line markers to carriage return/line feeds (CR/LF) as working text files are transferred to the Server's repository. When on, the working file's current end-of-line marker is converted to a CR/LF combination. When off, the default, no end-of-line conversion is performed. You would set this option to on, for example, when you add a working file to the repository from a computer running UNIX if you want the repository to store text files as CR/LF.
`-l`	Locks each file after it has been added to the view. Without `-l`, `-u`, or `-nel`, the files are unlocked by default.

`-mark`	Indicates that, if all the files are successfully added, the process item's status will be changed to fixed (for a change request), finished (for a task), or complete (for a requirement). The files are pinned to the revision with the new status.
`-nel`	Non-exclusively locks each file after it has been added.
`-ro`	Makes the working file read-only after this operation. Without this option, the file remains as it was prior to the operation. Usually, you use `-ro` to prevent yourself from editing a file that is not locked by you. `-ro` must be used with `-l` or `-u` or `-nel`. If you use `-ro`, you cannot use `-rw`.
`-rw`	Makes the working file read-write after this operation. Without this option, the file remains as it was prior to the operation. `-rw` must be used with `-l` or `-u` or `-nel`. If you use `-rw`, you cannot use `-ro`.
`-u`	Leaves the newly added files unlocked.
`-vl`	Specifies a label to be applied to the new files. The label is enclosed in double quotation marks. This option can appear in the command more than once. The label can be either a view or revision label, but it must already exist in the application.

Example

The following example uses `stcmd add` to add all `.doc` files with the status Not In View to User Manual, a child of the root folder StarDraw (in the StarDraw view of the StarDraw project). It locks the files and gives them the description "First draft of chapter".

```
stcmd add -p "JMarsh:password@Orion:1024/StarDraw/StarDraw/User Manual" -l-d "First draft
of chapter" "*.doc"
```

Related Reference

Client Command-line Operations

Add Folders: stcmd add-folder

Use `stcmd add-folder` to add StarTeam folders to a view from the command line. You can add the folder to the root folder or any other folder in that view. The working folder for your new StarTeam folder is created by default within StarTeam, not on your workstation. The working folder has the same name as the StarTeam folder. It is a child folder of the working folder for the StarTeam folder's parent.

For example, suppose you create a StarTeam folder named "Wizard". Wizard is a child of a StarTeam folder whose working folder is "C:\StarDraw". Therefore, Wizard's working folder becomes "C:\StarDraw\Wizard".

Using the `-is` option allows you to add a branch of folders to the project view's folder hierarchy. When you use `-is`, use either `-rp` or `-fp` to specify the folder on your workstation whose child folders will become the new StarTeam folder's child folders. Using `-fp` is recommended, as it specifies the path directly to the parent of those child folders. In contrast, `-rp`, which specifies the path to the working folder used for the view's root folder, appends StarTeam folder names in the hierarchy from the root folder to the new folder to the path you specify. Only when you use the `-is` option do `-rp` and `-fp` have any effect on this command.

Syntax

The syntax for this command is as follows:

```
stcmd add-folder -p "projectSpecifier" [-pwdfile "filePath"] [-cmp] [-csf] [-encrypt
encryptionType]
[-is] [-q] [-x] [-stop] [-rp "folderPath" | -fp "folderPath"] -name "folderName"
[-d "description"] [-ex "excludeType"] [-exlist "fileMask" | -exfile "fileName"]
```

Option	Description
`-d`	Specifies a description for the folder. Use a maximum of 254 characters.
`-ex`	Indicates the exclude lists to be used by this new folder. Exclude lists exclude certain files or types of files from visibility. If a working file in this folder's working folder would have the status Not In View but it matches a file specification in one of the exclude lists, the application does not display it at all. It is as though the file did not exist. For example, suppose you are creating files in an application that makes automatic backup copies of each file (with the extension .bak) every time you save a file. Your working folder might contain several .bak files, but you have no reason to add them to the project view. From the application, it is annoying to see these .bak files as possible candidates, so you exclude them. Excluding files is done on a per-folder basis. However, exclude lists can be inherited from parent folders. The full syntax is: `-ex excludeType` The types are: `inherit`, which indicates that this folder will inherit any exclude lists used by its parent folder and use the exclude list specified with either `-exfile or -exlist` (if one is created). This is the default. `local` , which indicates that this folder will use only the exclude list specified with either`-exfile or -exlist`. `none`, which indicates that this folder will use no exclude lists, regardless of what you specify with either `-exfile or -exlist`.
`-exfile`	Specifies the path to the file that contains the local exclude list for this folder. See `-exlist` for a description of the exclude list's contents.
`-exlist`	Specifies the local exclude list for this folder. Use a maximum of 254 characters. Enter one or more file specifications (using the standard * and ? wild cards), separated by commas, spaces, or semicolons. To include a comma, space, or semicolon as part of the specification, enclose the specification in double quotation marks. For example: `*.exe,*.dll p*z.doc;*.t?t "test *.*"`

	If you are using double-quotation marks in your exclude list or have a lengthy exclude list, we recommend that you use the -exfile option. With `-exlist`, each quotation mark in the exclude list needs to be preceded by the escape character for your system or shell. For example, the caret (^) works on NT systems. With`-exfile` , you do not need to use escape characters.
`-name`	Specifies a name for the folder. Use a maximum of 254 characters. In a file, if the exclude list contains double quotation marks, the escape character is unnecessary.

Example

The following example uses `stcmd add-folder` to create a folder named Wizard as a child of the StarDraw folder, the root folder of the StarDraw project view. In addition, it sets a local exclude list for Wizard. By default, Wizard inherits its parent folder's exclude lists and use the local one as well.

```
stcmd add-folder -p "JMarsh:password@Orion:1024/StarDraw/StarDraw/" -name "Wizard" -d
"StarDraw setup wizard" -exlist "*.bak"
```

The next example creates the same folder as in the previous example. However, it includes child folders. In this case, the folder with the path "C:\Wizard" has child folders (Source, Spec, and Doc), all of which are added as StarTeam folders in addition to Wizard. All of the new folders (Wizard, Source, Spec, and Doc) will have the default working folders assigned to them automatically by the StarTeam server, regardless of the setting for `-fp`. Wizard will be the parent of Source, Spec, and Doc. StarDraw is the parent of Wizard.

```
stcmd add p "JMarsh:password@Orion:1024/StarDraw/ StarDraw/" -name "Wizard" -d "StarDraw
setup wizard" -is -fp "C:\Wizard" exlist "*.bak"
```

Related Reference

Client Command-line Operations

Add Projects: stcmd add-project

Use `stcmd add-project` to add a project to a StarTeam Server configuration from the command line. When a project is created, its root view and the root folder for the root view are also created. In this command, the `-rp` option specifies the working folder for that root folder.

Using `-is` allows you to use the working folder's child folders as the root folder's child folders in the StarTeam folder hierarchy.

Syntax

The syntax for this command is as follows:

```
stcmd add-project [-pwdfile "filePath" ] [-cmp] [-encrypt encryptionType] [-is] [-q] [-x]
[-stop] -s "serverName" -name "projectName" -rp "folderPath" [-d "description"]
[-kw "fileMask" |-kwfile "fileName"] [-ex "excludeType" ] [-exlist "fileMask" |-exfile
"fileName"]
```

Option	Description
`-d`	Specifies a description for the project. Use a maximum of 254 characters.
`-ex`	Indicates the exclude lists to be used by the project's root folder. Exclude lists exclude certain files or types of files from visibility. If a working file in this folder's working folder would have the status Not In View but it matches a file specification in one of the exclude lists, the application does not display it at all. It is as though the file did not exist. For example, suppose you are creating files in an application that makes automatic backup copies of each file (with the extension .bak) every time you save a file. Your working folder might contain several .bak files, but you have no reason to add them to the project view. From the application, it is annoying to see these .bak files as possible candidates, so you exclude them. Excluding files is done on a per-folder basis. However, exclude lists can be inherited from parent folders. The full syntax is: `-ex excludeType` The types are: `inherit` Indicates that the root folder will inherit any exclude lists used by its parent folder and use the exclude list specified with either `-exfile` or`-exlist` (if one is created). This is the default—even though the root folder has nothing to inherit. `local` Indicates that the root folder will use only the exclude list specified with either `-exfile or -exlist`. `none` Indicates that the root folder will use no exclude lists, regardless of what you specify with either `-exfile` or`-exlist`.
`-exfile`	Specifies the path to the file that contains the local exclude list for the root folder. See -exlist for a description of the exclude list's contents.
`-exlist`	Specifies the local exclude list for the root folder. Use a maximum of 254 characters. Enter one or more file specifications (using the standard * and ? wild cards), separated by commas, spaces, or semicolons. To include a comma, space, or semicolon as part of the specification, enclose the specification in double quotation marks. For example: `*.exe,*.dll p*z.doc;*.t?t "test *.*"` If you are using double-quotation marks in your exclude list or have a lengthy exclude list, we recommend that you use the -exfile option. With -exlist, each quotation mark in the exclude list needs to be preceded by the escape character for your system or shell. For example, the caret (^) works on NT systems. With -exfile, you do not need to use escape characters.
`-kw`	Specifies the file extensions with which you want to use keywords. Use a maximum of 254 characters. Enter one or more file specifications (using the standard * and ? wild cards), separated by commas, spaces, or semicolons.

	To include a comma, space, or semicolon as part of the specification, enclose the specification in double quotation marks. For example: `*.cpp,*.h p*z.doc;*.t?t "test *.*"` If you are using double-quotation marks in your keyword list or have a lengthy list, we recommend that you use the -kwfile option. With -kwlist, each quotation mark in the keyword list needs to be preceded by the escape character for your system or shell. For example, the caret (^) works on NT systems. With -kwfile, you do not need to use escape characters.
`-kwfile`	Specifies the path to the file containing the file extensions with which you want to use keywords.
`-name`	Specifies a name for the project. Use a maximum of 254 characters.

`-s`	Identifies the Server. The full syntax is:
	`-s "userName:password@host:portNumber"`
	For example:
	`-s "JMarsh:password@orion:49201"`
	If the user name is omitted, the current user name is used. The user name in the example is “JMarsh”
	If the password is omitted, the user is prompted to enter the password. The password in the example is “password”. If the host name is omitted, the default is localhost. The host name in the example is “orion”.
	The port number is required. The default port number, 49201, is used in the example.

Example

The following example uses `stcmd add-project` to create a project named Integrations on the computer named Orion. (Orion is running an instance of the StarTeam Server with a server configuration that uses port 1024.) This command creates the project, specifies that the data sent between workstations and the server should be compressed and encrypted, and gives the project a description.

```
stcmd add-project -s "JMarsh:password@Orion:1024" -cmp -encrypt "RC4" -name "Integrations"
-rp "C:\integrations" -d "integrations between our products and our partner’s products"
```

Related Reference

Client Command-line Operations

Add Views: stcmd add-view

Use `stcmd add-view` to add a view to a StarTeam server configuration from the command line. When the view is created, its parent view is the view specified with the `-p` option and its root folder is the folder specified with the `-p` option. In this command, the `-rp` option specifies the working folder for the root folder. Use the following options to create the following types of views:

- Use `-dr` to create a read/write reference view.
- Use `-dr -ro` to create a read-only reference view.
- Use `-dr -ba` to create a branching view in which the behavior of existing items is set to branch on change.
- Use `-dr -bn` to create a branching view in which the behavior of existing items is not set to branch on change.
- If you do not use `-dr`, a blank view is created.

Syntax

The syntax for this command is as follows:

```
stcmd add-view -p "projectSpecifier" [-pwdfile "filePath"] [-cmp] [-encrypt
encryptionType]
[-q] [-x] [-stop] -name "viewName" [-rp "folderPath"] [-d "description"] [-dr [-ro | -ba |
-bn
[-cfgl "labelName" | -cfgp "stateName" | -cfgd "asOfDate"]]]
```

Option	Description
`-ba`	When used with `-dr`, specifies a branching view in which the behavior of existing items is set to branch on change. The value of the view property Set Items Shared Into View To Branch On Change is initially set. This option must be used with `-dr`.
`-bn`	When used with `-dr`, specifies a branching view in which the behavior of existing items is not set to branch on change. The value of the view property Set Items Shared Into View To Branch On Change is initially cleared. This option must be used with `-dr`.
`-cfgd`	Configures the view as of the specified date/time. Examples include: "12/29/01 10:52 AM" "December 29, 2001 10:52:00 AM PST" "Monday, December 29, 2001 10:52:00 AM PST". This option must be used with one of the following combinations: `-dr -ro`, `-dr -ba`, or `-dr -bn`.
`-cfgl`	Configures the view using the specified label. Without `-cfgl` , `-cfgp`, or`-cfgd`, the view's current configuration is used. This option must be used with one of the following combinations: `-dr -ro`, `-dr -ba`, or`-dr -bn`.
`-cfgp`	Configures the view using the specified promotion state. This option must be used with one of the following combinations: `-dr -ro`, `-dr -ba`, or `-dr -bn`.
`-d`	Specifies a description for the view. Use a maximum of 254 characters.
`-dr`	Specifies a derived view. All views, except blank views are derived. See also `-ba`, `-bn`, and `-ro`. When used without `-ba`, `-bn`, or `-ro`, a read/write reference view is created. The configuration of a read/write reference view is the same configuration as its parent view. Therefore, using `-dr` without `-ba`, `-bn`, or `-ro`, but with `-cfgl`, `-cfgp`, or `-cfgd` results in an error message.

	When this option is not used, a blank view is created. For blank views, the value of the view property named Set Items Shared Into View To Branch On Change is initially cleared.
`-name`	Specifies a name for the view. Use a maximum of 254 characters.
`-ro`	When used with `-dr`, specifies a read-only reference view.

Examples

The following example uses `stcmd add-view` to create a branching view named Maintenance 5.1 on the computer named Orion. (Orion is running an instance of the StarTeam Server with a server configuration that uses port 1024.)

This command creates the view as a child of the existing StarDraw view and uses the StarDraw folder as its root folder. The new view is based on the label used for the last build of the 5.1 product before it shipped (Build 403). It has a working folder that is different from the parent's working folder. All existing items in the view will have their behavior set to branch on change.

```
stcmd add-view -p "JMarsh:password@Orion:1024/StarDraw/StarDraw/" -cmp -encrypt "RC4" -
name
"Maintenance 5.1" -rp "C:\StarDraw\Maintenance 5.1" -d "Maintenance view for 5.1 release of
our product"
-dr -ba -cfgl "Build 403"
```

The following example uses `stcmd add-view` to create a read/write reference view named Rooted At Source Code on the computer named Orion. This command creates the view as a child of the existing StarDraw view and uses the SourceCode folder as its root folder. It has the same working folder as its parent. Because a read/write reference view must have the same configuration as its parent, none of the `-cfgl`, `-cfgp`, and `-cfgd` options can be used.

```
stcmd add-view -p "JMarsh:password@Orion:1024/ StarDraw/StarDraw/Source Code"
-cmp -encrypt "RC4" -name "Rooted At SourceCode" -d
"StarDraw main view but with SourceCode folder as the root of the hierarchy" -dr
```

Related Reference

Client Command-line Operations

Apply Labels: stcmd apply-label

Use `stcmd apply-label` to label specified file revisions with view or revision labels. The labels must already exist in StarTeam. You can create the labels in StarTeam or with the `stcmd label` command.

Syntax

The syntax for this command is as follows:

```
stcmd apply-label -p "projectSpecifier" [-pwdfile "filePath"] [-cmp] [-csf]
[-encrypt encryptionType] [-is] [-q] [-x] [-stop] [-rp "folderPath" | -fp "folderPath"]
[-filter "fileStatus"] [-vl "labelName" | -vd "asOfDate" | -vn revisionNumber] -lbl
"labelName" [files...]
```

Option	Description
`-filter`	Specifies a string of one or more characters, each of which represents a file status. Never include spaces or other whitespace in this string. The label is applied only to the files that currently have the specified statuses. You cannot apply labels to files that are Not In View. The letters used to represent the statuses are: C for Current, M for Modified, G for Merge, O for Out of Date, I for Missing, and U for Unknown.
`-lbl`	Specifies the label name to be added to the specified revisions. This option can be used more than once. The application attaches all the labels to the specified file or revisions.
`-vd`	Specifies the as-of date/time used to identify the revisions that get the new label. Examples include: "12/29/01 12:41 PM" "December 29, 2001 12:41:21 PM" "Monday, December 29, 2001 12:41"
`-vl`	Specifies the revision or view label used to identify the revisions that get the new label. This label must already exist in the application. Without the `-vn` or `-vd` or `-vl` option, the tip revision of each file receives the label.
`-vn`	Specifies the revision number used to identify the revisions that get the new label.

Examples

The following example uses `stcmd apply-label` to apply the label Beta to files in User Manual, a child of the root folder StarDraw (in the StarDraw view of the StarDraw project). StarTeam applies the label to the revisions of those files that were current at noon on July 7, 2003.

```
stcmd apply-label -p "JMarsh:password@Orion:1024/StarDraw/StarDraw/User Manual" -vd
"07/07/03 12:00 PM" -lbl "Beta" "*"
```

Related Reference

Client Command-line Operations

Check In Files: stcmd ci

Use `stcmd ci` to check files into a StarTeam repository (or vault) from a working folder using the command line.

You can simultaneously link the new file revisions to a process item. All the files successfully added using this command will be linked and pinned to the tip revision of the process item. Use the `-active` option to specify the currently active process item (previously set using a StarTeam client on your workstation).

If no item is active or you prefer to use another item, use the option that indicates the type of the process item (`-cr, -req`, or `-task`), followed by the complete path from the root folder of the StarTeam project view to the item, using the forward slash (/) as a delimiter between folder names. For out-of-view process items, specify the project name and view name in front of the complete folder path. Separate the view path with a colon (:). For example, `-cr MyProject/RootView:ChildView/SourceCode/37` specifies change request 37 in the `SourceCode` folder of the `ChildView` view in the `MyProject` project. During execution, the process first assumes that the process item is in the current view, and it checks the current view to determine whether the full path corresponds to a folder path within that view. If the process item is not found in the current view, it is treated as an out-of-process item, and the search for the process item begins from the project and view.

Use the `-mark` option to simultaneously mark the process item as fixed, finished, or complete, depending on its type. For example, a change request can be marked as fixed. The item is not marked as fixed, finished, or complete unless all the files are successfully added.

Syntax

The syntax for this command is as follows:

```
stcmd ci -p "projectSpecifier" [-pwdfile "filePath"] [-cmp] [-csf] [-encrypt
encryptionType]
[-is] [-q] [-x] [-stop] [-rp "folderPath" | -fp "folderPath"] [-filter "fileStatus"] [-l |
-u | -nel]
[-ro | -rw]] [-vl "labelName"] [-nomove] [-f NCI] [-o | -i ] [-r "comment" | -rf " fileName
"] [-eol
[on | off]] [[ -active | [-cr | -req | -task ] processItemPath] [-mark]] [files...]
```

Option	Description
`-active`	The active process item.

`-cr`
`-req`
`-task`
	Complete path from the project view's root folder to the change request, requirement, or task number to be used as a process item. Use the forward slash (/) as a delimiter between folder names.
	For out-of-view process items, specify the project name and view name in front of the complete folder path. Separate the view path with a colon (:). For example, `-cr MyProject/RootView:ChildView/SourceCode/37` specifies change request 37 in the `SourceCode` folder of the `ChildView` view in the `MyProject` project.
	During execution, the process first assumes that the process item is in the current view, and it checks the current view to determine whether the full path corresponds to a folder path within that view. If the process item is not found in the current view, it is treated as an out-of-process item, and the search for the process item begins from the project and view.

`-eol`	Automatically converts end-of-line markers to carriage return/line feeds (CR/LF) as working text files are transferred to the Server's repository. When on, the working file's current end-of-line marker is converted to a carriage return/line feed (CR/LF) combination. When off, the default, no end-of-line conversion is performed. For Windows clients, the end-of-line marker is CR/LF; for UNIX platforms, it is a line feed (LF). You would set this option to on, for example, when you check a working file into the repository from a computer running UNIX and the repository stores text files as CR/LF.
`-f NCI`	Specifies the check-in of any file whose status is Modified. NCI stands for "needs check-in." No other types of files are selected for check-in.

	`-f NCI` is ignored if `-filter`is used.
`-filter`	Specifies a string of one or more characters, each of which represents a file status. Never include spaces or other white space in this string. Only files that currently have the specified statuses will be checked in. You cannot check in files that are Not In View. The letters used to represent the statuses are: C for Current, M for Modified, G for Merge, O for Out of Date, I for Missing, and U for Unknown. `-filter` takes precedence over `-f NCI`. If you use G, O, or U, you must also specify`-i`or `-o`. Otherwise, the G, O, or U is ignored.
`-i`	Allows an interactive check-in for files whose status would normally not allow them to be checked in. You are asked about each file whose status is Merge, Out of Date or Unknown. You can force the file to be checked in with your response. If you use the `-i` option, you cannot use the `-o` option.
`-l`	Locks each file after it has been checked in. Without `-l`,`-u`, or `-nel`, the files lock status is unchanged.
`-mark`	Indicates that, if all the files are successfully added, the process item's status will be changed to fixed (for a change request), finished (for a task), or complete (for a requirement). The files are pinned to the revision with the new status.
`-nel`	Non-exclusively locks each file after it has been checked in.
`-nomove`	Stops the application of the label specified by the -vl option if the file, which is being checked in, already has a revision with that label. Otherwise, the label will be moved from the currently labeled revision to the newly checked-in revision.
`-o`	Forces check-in for files whose status would normally not allow them to be checked in. This option forces all files whose status is Merge, Out of Date or Unknown to be checked in. If you use the -o option, you cannot use the -i option.
`-r`	Provides a revision comment, usually the reason for checking in the files. If you use the `-r` option, you cannot use the `-rf` option.
`-rf`	Provides the path to the file that contains the revision comment. If you use the `-rf` option, you cannot use the`-r` option.
`-ro`	Makes the working file read-only after this operation. Without this option, the file remains as it was prior to the operation. Usually, you use `-ro` to prevent yourself from editing a file that is not locked by you. `-ro` must be used with `-l` or `-u` or `-nel`. If you use `-ro`, you cannot use `-rw`.
`-rw`	Makes the working file read-write after this operation. Without this option, the file remains as it was prior to the operation. `-rw` must be used with `-l` or `-u` or `-nel`. If you use `-rw`, you cannot use `-ro`.
`-u`	Unlocks the newly checked-in files.
`-vl`	Specifies a label (created using `stcmd label`) to be applied to the checked-in files. The label is enclosed in double quotation marks. This option can appear in the command more than once. The label can be either a view or revision label, but it must already exist in the application.

Example

The following example uses `stcmd ci` to check in .bmp files to Online Help, a child of the root folder StarDraw (in the StarDraw view of the StarDraw project). The command unlocks the files, makes the working copy read only, and gives the files a revision comment (usually a reason for checking in the files).

```
stcmd ci -p "JMarsh:password@Orion:1024/StarDraw/StarDraw/SourceCode/Online Help" -u -ro -
r
"revised for beta" "*.bmp"
```

Related Reference

Client Command-line Operations

Check Out Files: stcmd co

Use `stcmd co` to check out files from a StarTeam repository (or vault) to your working folder using the command line. Unless you use `-o`, this command pauses at each file with a Modified, Merge or Unknown status to let you know that the file will not be checked out.

With the `-merge` option, you can merge files as part of the check-out process. Merging is not part of the check-in process.

Syntax

The syntax for this command is as follows:

```
stcmd co -p "project" [-pwdfile "filename"] [-epwdfile "filename"] [-cmp] [-encrypt RC4,
RC2_ECB, RC2_CBC, RC2_CFB]
[-cfgl "label" | -cfgp "promotion state" | -cfgd "date"] [-is] [—csf] [-q] [-x] [-stop]
[-rp "directory" | -fp "directory"] [-filter "filter"][-?] [-h] [nologo] [-o | -i | -merge]
[-hook "executable"]
[-l | -u | -nel] [-ro | -rw]] [-vl "name" | -vd "date" | -vn number]
[-f NCO]  [-ts] [-eol [on | off | cr| lf| crlf ]] [-fs] [-dryrun | —alwaysprompt | —
neverprompt | —conflictprompt] [-mpxCacheAgent number]
[—useMPXCacheAgent “host”port” | autolocate]]  [files...]
```

Option	Description
`-cfgd`	Configures the view as of the specified date/time. Examples include: "12/29/01 10:52 AM" "December 29, 2001 10:52:00 AM PST" "Monday, December 29, 2001 10:52:00 AM PST"
`-cfgl`	Configures the view using the specified label. Without `-cfgl`, `-cfgp`, or `-cfgd`, the view's current configuration is used.
`-cfgp`	Configures the view using the specified promotion state.
`-eol`	Can automatically convert end-of-line markers. When on, text files are transferred from the Server's repository to the workstation's working folder with the end-of-line convention for the platform executing the command as determined by the Java VM. When off, the default, no end-of-line conversion is performed. Using off is the same as not using -eol at all. When you specify the end-of-line character (cr, lf, or crlf), text files are transferred from the Server's repository to the workstation's working folder with the specified end-of-line convention. For Windows clients, the end-of-line marker is a carriage return/line feed (crlf) combination; for UNIX platforms, it is a line feed (lf); for MAC systems, a carriage return (cr). You would set this option to on or lf, for example, when you compare a file from the repository and a working file on a UNIX system (if the repository stores text files as crlf).
`-f NCO`	Specifies the check-out of any file whose status is Missing or Out of Date. NCO stands for “needs check-out.” No other files are selected for check-out. `-f NCO` is ignored if `-filter` is used.
`-filter`	Specifies a string of one or more characters, each of which represents a file status. Never include spaces or other whitespace in this string. Only files that currently have the specified statuses will be checked out. You cannot check out files that are Not In View. The letters used to represent the statuses are: C for Current, M for Modified, G for Merge, O for Out of Date, I for Missing, and U for Unknown.

	`-filter` takes precedence over `-f NCO`. If you use G, M, O, or U, you must also specify `-merge` or `-o` to force the check-out operation. Otherwise, the G, M, O, or U is ignored.
`-fs`	Prevents file statuses from being remembered after the check-out occurs. Subsequent status values for these files will be incorrect and indeterminate. Use this option where a file's status is irrelevant. For example, if you routinely delete the working folders before checking out files for a build, there are no files and their statuses do not matter. Be aware that the file statuses may never be known—even if you use the update-status command later. You can do a force check out without the -fs option to obtain current files with correct statuses.
`-hook`	Used only with `-merge`. Enables you to specify an alternate application (other than that available with the application) to perform the merge. The value of the option should be the name of a program to run to perform the merge, for example: `-hook mymerge.sh` The merge application must return an exit code of 0 meaning that no conflicts were detected and an exit code of 1 meaning that conflicts were detected. Any other value indicates an error. The merge application cannot be a batch file on Windows platforms because, when run via Java, the result code is not returned properly. For each file to be merged, stcmd provides three arguments to the merge application. The arguments are the fully qualified paths to the following three files (in this order): localFile: The local working file commonFile: The contents of the file revision in the repository which is the most recent common version between the file being modified locally and the specified revision in the repository otherFile: The contents of the file revision being checked out The merge hook program must send the merge result to standard output (stdout).
`-i`	Allows an interactive check-out for files whose status would normally not allow them to be checked out. You are asked about each file whose status is Modified, Merge, or Unknown. You can force the file to be checked out with your response. If you use the `-i` option, you cannot use the `-o` or `-merge` option.
`-l`	Locks each file after it has been checked out. If `-l` or `-u` or `-nel` is not used, the files lock status remains unchanged.
`-merge`	Enables you to merge the working file with the revision being checked out and with the revision upon which both of these is based. The working file must have the status Merge. You can specify one of the following reporting options with `-merge`: `-dryrun`: Indicates whether the merged results file has conflicts or not; the local working file is unchanged as nothing is checked out. This provides a preview. `-alwaysprompt`: Always prompts the user to save the merged result file in the working folder whether or not there were merge conflicts. `-neverprompt`: Always saves the merged results file to the working folder. `-conflictprompt`: Prompts the user to save the merged results file only if conflicts were detected. The options `-dryrun`, `-alwaysprompt`, `-neverprompt`, and `-conflictprompt` are mutually exclusive. When none are specified, the default behavior is `-conflictprompt`. You can specify an alternate application (other than that available with the application) to perform the merge. If you use the `-merge` option, you cannot use the `-i` or `-o` option. If you are not using `-hook`, and you save a merged file with conflicts, each conflict is marked inside the working file as follows: <<<<<<< fileName (local) line as it appears in the local file ======= line as it appears in the revision being checked out

	>>>>>>> fileName (version x) where fileName is the name of the file and x is the number of the revision being checked-out. If you do not specify `-merge`, files with the status Merge are ignored unless you specify`-o` and force the check-out operation.
`-nel`	Non-exclusively locks the file after it has been checked out.
`-o`	Forces the check-out of files of any status. If you use the `-o` option, you cannot use the `-i` or `-merge` option.
`-ro`	Makes the working file read-only after this operation. Without this option, the file remains as it was prior to the operation. Usually, you use `-ro` to prevent yourself from editing a file that is not locked by you. `-ro` must be used with `-l` or `-u` or `-nel`. If you use `-ro`, you cannot use `-rw`.
`-rw`	Makes the working file read-write after this operation. Without this option, the file remains as it was prior to the operation. `-rw` must be used with `-l` or `-u` or `-nel`. If you use `-rw`, you cannot use `-ro`.
`-ts`	Sets each working file's time stamp to the check-out time. Without this option, the file is given the same time stamp as the checked-in revision of the file.
`-u`	Unlocks the checked-out files.
`-vd`	Specifies the as-of date/time used to identify the revisions to be checked out. The last revision before the specified date/time is the one checked out for each file. See the date/time examples for `-cfgd` above.
`-vl`	Specifies the revision or view label used to identify the revisions to be checked out. Without the`-vn` or `-vd` or `-vl` option, the tip revision of each file is checked out.
`-vn`	Specifies the revision number of the files to be checked out.

Examples

The following example uses `stcmd co` to lock and check out .doc files from User Manual, a child of the root folder StarDraw (in the StarDraw view of the StarDraw project).

```
stcmd co -p "JMarsh:password@Orion:1024/StarDraw/StarDraw/User Manual" -l "*.doc"
```

The next example uses stcmd co to merge a readme file.

```
stcmd co -p "NTesla:@10.50.5.179:49201/WebDev/WebDev" -encrypt RC4 -fp
"/export/home0/johnson/working" -merge "README"
```

Related Reference

Client Command-line Operations

Compare File Revisions: stcmd diff

Use `stcmd diff` to display differences between two revisions of a file. The command can be applied to more than one file. If you do not specify any revisions using `-vn`, `-vd`, or `-vl`, the working copy of each specified file is compared to the tip revision in the repository (or vault) for this file. If you specify a single revision, the working copy of each specified file is compared to that revision. If you specify two revisions, those two revisions of each specified file are compared.

When comparing text files, the differences can be displays. When comparing binary files, output results indicate whether the revisions of the file are the same or different.

Syntax

The syntax for this command is as follows:

```
stcmd diff -p "projectSpecifier" [-pwdfile "filePath"] [-cmp] [-csf]
[-encrypt encryptionType] [-cfgl "labelName"| -cfgp "stateName" | -cfgd "asOfDate"]
[-is] [-q] [-x] [-stop] [-rp "folderPath" | -fp "folderPath"] [-filter "fileStatus"]
[-eol [on | off | cr| lf| crlf]] [-w | -Bpvcs | -b] [-i] [-m "maskSet"] [-t number] [-c
number] [-n]
[-nd] [-e] [-vl "labelName" | -vd "asOfDate" | -vn revisionNumber] [files...]
```

Option	Description
`-b`	When comparing two lines of text files, ignores trailing whitespace and treats all other strings of whitespace as equal in length. For example, the following lines are equivalent: " hi mom " " hi mom"
`-Bpvcs`	When comparing two lines of text files, ignores leading and trailing whitespace. For example, the following lines are equivalent because there is only one space between "hi" and "mom": " hi mom " " hi mom" but the next line is not equivalent: "hi mom"
`-c`	Specifies the number of unchanged lines to display before and after a difference is found in text files. Without this option, all lines of the files are displayed. For example, `-c 2` places two unchanged lines before and after each line or set of lines that has changed.
`-cfgd`	Configures the view as of the specified date/time. Examples include: "12/29/01 10:52 AM" "December 29, 2001 10:52:00 AM PST" "Monday, December 29, 2001 10:52:00 AM PST"
`-cfgl`	Configures the view using the specified label. Without `-cfgl`, `-cfgp`, or `-cfgd`, the view's current configuration is used.
`-cfgp`	Configures the view using the specified promotion state.
`-e`	Causes the command to return the following exit codes: `0` if all the compared files are equivalent `1` if an error condition occurs `2` if at least one file is different

	Use `-e` with either text or binary files.
`-eol`	This command is irrelevant at this point in time because `diff` currently ignores end-of-line markers. If two lines are the same except for this, they are reported to be identical.
`-filter`	Specifies a string of one or more characters, each of which represents a file status. Never include spaces or other white space in this string. Only files that currently have the specified statuses will be compared. You cannot compare revisions of files that are Not In View. The letters used to represent the statuses are: `C` for Current, `M` for Modified, `G` for Merge, `O` for Out of Date, **I** for Missing, and `U` for Unknown.
`-i`	Ignores the case of letters when comparing two text files. For example, "A" is equivalent to "a".

`-m`	When comparing two text files, ignores the characters in certain columns as specified by one or more masks. Each mask has the following syntax:
	
	`"columnNumber-columnNumber[(numeric)]"`
	
	For example, "1-6" ignores the characters in the first six columns of each line, and "1-6 (numeric)" ignores the first six columns of each line if the character in column 1 is a digit in both files.
	
	You can use a series of masks, but they must be separated by commas. The syntax is:
	
	`"mask[,mask]..."`

`-n`	Suppresses the display of line numbers in the two text files.
`-nd`	Suppresses the display of differences in two text files. Comparisons of binary files do not display differences. This option is useful with the`-e` option.
`-t`	Specifies the number of spaces to use for each tab stop when displaying the file differences for text files. The default is four. Use `-t 0` to suppress tab conversion.
`-vd`	Specifies the as-of date/time used to identify the revisions to be compared. The last revision before the specified date/time is the one used. See the date/time examples for `-cfgd` above.
`-vl`	Specifies the revision or view label used to identify the revisions to be compared. You can specify any combination of zero, one, or two of the `-vn`, `-vd`, or `-vl` options. Use zero options to compare the working file to the tip revision, one to compare the working file to the specified revision, and two to compare two revisions.
`-vn`	Specifies the revision number to be compared.

`-w`	Ignores all whitespace (tabs and spaces) when comparing two lines in text files. For example, the following lines would be equivalent:
	
	`" a = ( b + 2);"`
	
	`"a=(b+2);"`
	
	The`-w`, `-Bpvcs`, and `-b` options are mutually exclusive.

Example

The following example uses `stcmd diff` to compare the Beta1 and Beta2 revisions of each of the `.cpp` files in the folder `SourceCode`, a child of the root folder `StarDraw` (in the StarDraw view of the StarDraw project). It ignores all white space.

```
stcmd diff -p "JMarsh:password@Orion:1024/StarDraw/StarDraw/SourceCode" -w -vl
"Beta1" -vl "Beta2" "*.cpp"
```

Related Reference

Client Command-line Operations

Change File Descriptions: stcmd dsc

Use `stcmd dsc` to change a file description from the command line. This command creates a new file revision with new description as one of its properties.

Syntax

The syntax for this command is as follows:

```
stcmd dsc -p "projectSpecifier" [-pwdfile "filePath"] [-cmp] [-csf] [-encrypt
encryptionType]
[-is] [-q] [-x] [-stop] [-rp "folderPath" | -fp "folderPath"] [-filter "fileStatus"] -d
"description" [files...]
```

Option	Description
`-d`	Provides a description for the files. The description is enclosed in double quotation marks.
`-filter`	Specifies a string of one or more characters, each of which represents a file status. Never include spaces or other whitespace in this string. Only files that currently have the specified statuses will be given the description. You cannot change the descriptions of files that are Not In View. The letters used to represent the statuses are: C for Current, M for Modified, G for Merge, O for Out of Date, I for Missing, and U for Unknown.

Example

The following example uses `stcmd dsc` to change the description of stdafx.cpp in User Manual, a child of the root folder StarDraw (in the StarDraw view of the StarDraw project).

```
stcmd dsc -p "JMarsh:password@Orion:1024/StarDraw/StarDraw/SourceCode" -d
"SourceCode for StarTeam" "stdafx.cpp"
```

Related Reference

Client Command-line Operations

Create Labels: stcmd label

Use `stcmd label` to create a view or revision label. A view label can be designated as a build label. By default, view labels are automatically applied to every folder, file, change request, requirement, topic, and task in the view. By default, revision labels are not applied to any items.

You can use `stcmd apply-label` to apply labels created with `stcmd label` to specified files. You can also use the label option (`-vl`) in `stcmd ci` to attach your new label to files as you check them in.

Option	Description
`-b`	Specifies that the new label is a build label. Without either`-b` or `-r`, the label is a view label. View labels (and a build label is a special type of view label) are immediately and automatically applied to every folder, file, change request, task, and topic in the view.
`-d`	Specifies the description of the label.
`-f`	Creates the new label as a frozen label.
`-nl`	Specifies the new label's name.
`-r`	Specifies that the new label is a revision label. You can use the new label to label files that you check in. This command does not attach the new label to any items unless you create the label by copying an existing revision label that is attached to one or more items. See the `-vl` option below.
`-vd`	Specifies the as-of date/time for the label when you create a view label. -vd is ignored if you are creating a revision label. Without the `-vn`, or `-vd`, or `-vl` option, the current time is used for view labels. Examples include: "12/29/01 12:41 PM" "December 29, 2001 12:41:21 PM" "Monday, December 29, 2001 12:41"
`-vl`	Specifies the name of the label to be copied. The name of the label is enclosed in double quotation marks. The specified label must already exist in the application and must be the same type as the label you are creating. If the specified label is a revision label that is attached to one or more items, the new label will be attached to those same items. Without the `-vn`, or `-vd`, or`-vl` option, the current time is used for view labels. If this is a revision label, make sure that you have set `-r`.
`-vp`	Specifies a promotion state whose label will be copied. The name of the state is enclosed in double quotation marks. The specified label must already exist in the application, and you must be creating a view label for this option to be valid. Without the `-vn`, or `-vd`, or `-vl` option, the current time is used for view labels. You can create a view label based on a promotion state ONLY when the promotion state is set to a label and not set to `Current`.

Syntax

The syntax for this command is as follows:

```
stcmd label -p "username:password@host:port/project/view/folder" [-pwdfile "filePath"] [-
cmp] [-encrypt encryptionType]
[-q] [-x] [-stop] -nl "labelName" [-vl "labelName" | -vd "asOfDate" | -vp stateName]
[-d "description"] [-b | -r] [-f]
```

Example

The following example uses stcmd label to create a new build label named Beta for the StarDraw view of the StarDraw project.

```
stcmd label -p "JMarsh:password@Orion:1024/StarDraw/StarDraw" -nl "Beta" -b
```

Related Reference

Client Command-line Operations

Create Working Folders: stcmd local-mkdir

Use `stcmd local-mkdir` to create the working folder or working directory on your workstation for the specified StarTeam folder. Use -is to create working folders (or working directories) for the specified StarTeam folder's child folders as well.

Syntax

The syntax for this command is as follows:

```
stcmd local-mkdir -p "projectSpecifier" [-pwdfile "filePath"] [-cmp] [-csf]
[-encrypt encryptionType] [-cfgl "labelName" | -cfgp "stateName" | -cfgd "asOfDate"]
[-is] [-q] [-x] [-stop] [-rp "folderPath" | -fp "folderPath"]
```

Option	Description
`-cfgd`	Configures the view as of the specified date/time. Examples include: "12/29/01 10:52 AM" "December 29, 2001 10:52:00 AM PST" "Monday, December 29, 2001 10:52:00 AM PST"
`-cfgl`	Configures the view using the specified label. Without `-cfgl`, `-cfgp`, or `-cfgd`, the view's current configuration is used.
`-cfgp`	Configures the view using the specified promotion state.

Example

The following example uses `stcmd local-mkdir` to create the working folders for SourceCode, a child of the root folder StarDraw (in the StarDraw view of the StarDraw project) and its child folders.

```
stcmd local-mkdir -p "JMarsh:password@Orion:1024/ StarDraw/StarDraw/SourceCode" -is
```

Related Reference

Client Command-line Operations

Delete Local Files: stcmd delete-local

Use `stcmd delete-local` to delete files from a working folder. You can delete files that are under version control, as well as files that are not in StarTeam. This action does not remove any files from version control. It merely reduces the amount of data stored on your workstation in a working folder. If you are deleting files based on their StarTeam status, it is a good idea to use `stcmd update-status` first.

Syntax

The syntax for this command is as follows:

```
stcmd delete-local -p "projectSpecifier" [-pwdfile "filePath"] [-cmp] [-csf]
[-encrypt encryptionType] [-is] [-q] [-x] [-stop] [-rp "folderPath" | -fp "folderPath"]
[-cfgl "labelName" | -cfgp "stateName" | -cfgd "asOfDate"] [-filter "fileStatus"] [files...]
```

Option	Description
`-cfgd`	Configures the view as of the specified date/time. Examples include: "12/29/01 10:52 AM" "December 29, 2001 10:52:00 AM PST" "Monday, December 29, 2001 10:52:00 AM PST"
`-cfgl`	Configures the view using the specified label. Without `-cfgl`, `-cfgp`, or `-cfgd`, the view's current configuration is used.
`-cfgp`	Configures the view using the specified promotion state.
`-filter`	Specifies a string of one or more characters, each of which represents a file status. Never include spaces or other whitespace in this string. Only files that currently have the specified statuses will be deleted. The letters used to represent the statuses are: `C` for Current, `M` for Modified, `G` for Merge, `O` for Out of Date, `I` for Missing, and `U` for Unknown.

Example

The following example uses `stcmd delete-local` to delete some files from the working folder for the StarTeam folder named `SourceCode`. `SourceCode` is a child of the root folder `StarDraw` (in the StarDraw view of the StarDraw project). This example deletes all files that are not under version control. Those files have the file status **Not In View**.

```
stcmd delete-local -p "JMarsh:password@Orion:1024/StarDraw/StarDraw/SourceCode"
-filter "N" "*"
```

Related Reference

Client Command-line Operations

Display File History: stcmd hist

Use `stcmd hist` to display the revision history of files.

Syntax

The syntax for this command is as follows:

```
stcmd hist -p "projectSpecifier" [-pwdfile "filePath"] [-cmp] [-csf]
[-encrypt encryptionType] [-cfgl "labelName" | -cfgp "stateName" | -cfgd "asOfDate"]
[-is] [-q] [-x] [-stop] [-rp "folderPath" | -fp "folderPath"] [-filter "fileStatus"]
[files...]
```

Option	Description
`-cfgd`	Configures the view as of the specified date/time. Examples include: "12/29/01 10:52 AM" "December 29, 2001 10:52:00 AM PST" "Monday, December 29, 2001 10:52:00 AM PST"
`-cfgl`	Configures the view using the specified label. Without `-cfgl`, `-cfgp`, or `-cfgd`, the view's current configuration is used.
`-cfgp`	Configures the view using the specified promotion state.
`-filter`	Specifies a string of one or more characters, each of which represents a file status. Never include spaces or other whitespace in this string. Only files that currently have the specified statuses will be reported. You cannot review the history of files that are Not In View. The letters used to represent the statuses are: C for Current, M for Modified, G for Merge, O for Out of Date, I for Missing, and U for Unknown.

Example

The following example uses `stcmd hist` to display the revision history of the file `star.h` in `SourceCode`, a child of the root folder `StarDraw` (in the StarDraw view of the StarDraw project).

```
stcmd hist -p "JMarsh:password@Orion:1024/StarDraw/StarDraw/SourceCode" "star.h"
```

Related Reference

Client Command-line Operations

List Files: stcmd list

Use`stcmd list` to list all files in the folder specified by the `-p` option. The files are those that existed at a specific time or that have a specific label.

Syntax

The syntax for this command is as follows:

```
stcmd list -p "projectSpecifier" [-pwdfile "filePath"] [-cmp] [-csf]
[-encrypt encryptionType] [-cfgl "labelName" | -cfgp "stateName" | -cfgd "asOfDate"]
[-is] [-q] [-x] [-stop] [-rp "folderPath" | -fp "folderPath"] [-filter "fileStatus"]
[-cf] [files...]
```

Option	Description
`-cf`	Causes the names of the child folders within the folder to be added to the list.
`-cfgd`	Configures the view as of the specified date/time. Examples include: "12/29/01 10:52 AM" "December 29, 2001 10:52:00 AM PST" "Monday, December 29, 2001 10:52:00 AM PST"
`-cfgl`	Configures the view using the specified label. Without `-cfgl`, `-cfgp`, or `-cfgd`, the view's current configuration is used. `-cfgp` configures the view using the specified promotion state.
`-filter`	Specifies a string of one or more characters, each of which represents a file status. Never include spaces or other white space in this string. Only files that currently have the specified statuses will be listed. You cannot list files that are Not In View. The letters used to represent the statuses are: C for Current, M for Modified, G for Merge, O for Out of Date, I for Missing, and U for Unknown.
`-short`	Provides a short and simple listing of local files and their statuses consisting of the abbreviation for the status and the relative path to the working file, for example: M /starteam/Server.java N /starteam/LabelInfo.java Without this option, the listing consists of: A line for each folder name followed by its working folder's path. Within a folder, a line for each file starting with the unabbreviated status and containing the rights, time stamp, and name of the file. For example: Folder: Source (working dir: E:\Source) Unknown rw 4/6/02 7:42:18 PM PST 230 req.bmp

Example

The following example uses `stcmd list` to list all files in SourceCode, a child of the root folder StarDraw (in the StarDraw view of the StarDraw project), as well as all the files in child folders of SourceCode.

```
stcmd list -p "JMarsh:password@Orion:1024/StarDraw/StarDraw/SourceCode" -is "*"
```

The `-short` provides a short , simple listing of local files and their statuses consisting of the abbreviation for the status and the relative path to the working file, for example:

`M /starteam/Server.java N /starteam/LabelInfo.java`.

Without the `-short` option, the listing consists of a line for each folder name followed by its working folder's path. Within a folder, the listing contains a line for each file starting with the unabbreviated status and containing the rights, time stamp, and name of the file. For example:

```
Folder: Source  (working dir: E:\Source) Unknown rw 4/6/02 7:42:18 PM PST        230 req.bmp
```

Related Reference

Client Command-line Operations

Lock and Unlock Files: stcmd lck

Use `stcmd lck` to lock or unlock files from the command line.

Option	Description
`-break`	Breaks the current lock by another user if you have the access rights to break locks.
`-filter`	Specifies a string of one or more characters, each of which represents a file status. Never include spaces or other white space in this string. Only files that currently have the specified statuses will be locked or unlocked. You cannot lock or unlock files that are Not In View. The letters used to represent the statuses are: C for Current, M for Modified, G for Merge, O for Out of Date, I for Missing, and U for Unknown.
`-l`	Locks the files. This is the default when `-l`, `-nel` or `-u` is not used.
`-nel`	Non-exclusively locks the files.
`-ro`	Makes the working file read-only after this operation. Without this option, the file remains as it was prior to the operation. Usually, you use -ro to prevent yourself from editing a file that is not locked by you. -ro must be used with `-l` or `-u` or `-nel`. If you use `-ro`, you cannot use `-rw`.
`-rw`	Makes the working file read-write after this operation. Without this option, the file remains as it was prior to the operation. `-rw` must be used with `-l` or `-u` or `-nel`. If you use `-rw`, you cannot use `-ro`.
`-u`	Unlocks the files.

Syntax

The syntax for this command is as follows:

```
stcmd lck -p "projectSpecifier" [-pwdfile "filePath"] [-cmp] [-csf]
[-encrypt encryptionType] [-is] [-q] [-x] [-stop] [-rp "folderPath" | -fp "folderPath"]
[-filter "fileStatus"] [-break] [-l |-u | -nel] [-ro | -rw]] [files...]
```

Example

The following example uses `stcmd lck` to unlock all files in SourceCode, a child of the root folder StarDraw (in the StarDraw view of the StarDraw project), as well as all files in child folders of SourceCode.

```
stcmd lck -p "JMarsh:password@Orion:1024/StarDraw/StarDraw/SourceCode" -is -u "*"
```

Related Reference

Client Command-line Operations

Lock and Unlock Server: stcmd server-mode

If you have appropriate access rights or privileges, you can use `stcmd server-mode` to lock and unlock a server configuration. Locking a server configuration limits access to it while you perform backup or other procedures. When the server is locked, only server administration commands are accepted. When the server configuration is unlocked, normal operations resume.

Note: If the user name is “StarTeam,” this command requests a password for the user, even if one has already been provided or if the user has a blank password.

Syntax

The syntax for this command is as follows:

```
stcmd server-mode [-pwdfile "filePath"] [-cmp] [-encrypt encryptionType] [-q] [-x]
[-stop] -s "serverName" -mode [lock | exlock | unlock]
```

Option	Description
`exlock`	Exclusively locks the server so that no one else can access it.
`lock`	Nonexclusively locks the server. Only administrative commands can be performed.
`-mode`	Indicates whether the server is to be locked, exclusively locked, or unlocked. If you use `-mode` lock, only server administration commands are accepted until the server is unlocked. For example, you might use this command while running a backup program. If you use `-mode exlock`, only you can access the server until it is unlocked. For example, you might do this when creating a custom field. Use `-mode unlock` to make the server available to users again.
`unlock`	Unlocks the server so that anyone with the appropriate access rights can access it.

`-s`	Specifies the server. The full syntax is:
	`-s "userName:password@hostName:portNumber"`
	For example:
	`-s "JMarsh:password@orion:49201"`
	If the user name is omitted, the current user name is used. The user name in the example is “JMarsh”
	If the password is omitted, the user is prompted to enter the password. The password in the example is “password”
	f the host name is omitted, the default is localhost. The host name in the example is “orion”. The port number is required. The default port number, 49201, is used in the example.

Example

The following example uses `stcmd server-mode` to lock the server using port 1024 on Orion.

```
stcmd server-mode -s "JMarsh:password@Orion:1024" -mode lock
```

Related Reference

Client Command-line Operations

Remove Files: stcmd remove

Use `stcmd remove` to remove files from version control. The specified files and their revision histories no longer appear in StarTeam unless you roll back the project view to a time before they were removed.

Syntax

The syntax for this command is as follows:

```
stcmd remove -p "projectSpecifier" [-pwdfile "filePath"] [-cmp] [-csf]
[-encrypt encryptionType] [-is] [-q] [-x] [-stop] [-rp "folderPath" | -fp "folderPath"]
[-filter "fileStatus"] [-df] [files...]
```

Option	Description
`-df`	Deletes the user's working file. Without this option the working file remains in the working folder on your workstation.
`-filter`	Specifies a string of one or more characters, each of which represents a file status. Never include spaces or other whitespace in this string. Only files that currently have the specified statuses will be removed. You cannot remove files that are Not In View from version control. The letters used to represent the statuses are: C for Current, M for Modified, G for Merge, O for Out of Date, I for Missing, and U for Unknown.

Example

The following example uses `stcmd remove` to remove all .hm files from SourceCode, a child of the root folder StarDraw (in the StarDraw view of the StarDraw project), as well as all files in child folders of SourceCode. It also deletes the working files.

```
stcmd remove -p "JMarsh:password@Orion:1024/StarDraw/StarDraw/SourceCode" -is -df "*.hm"
```

Related Reference

Client Command-line Operations

Set Personal Options: stcmd set-personal-options

Use`stcmd set-personal-options` to set and list personal options. At this time, the only personal options that can be set with this command are:

- How file status information will be stored.
- Where file status information will be stored, when it is stored at a central location.

File status information is stored in the `starteam-client-options.xml` file. If no `starteam-client-options.xml` file exists, this command creates it in a default location. The default location is the same as the default for the central-status option (see below).

Syntax

The syntax for this command is as follows:

```
stcmd set-personal-options [-q] [-x] [-stop] [-central-status | -per-folder-status]
[-central-repository "folderPath"] [-list]
```

Option	Description
`-central-repository`	Enables you to specify a location for the central repository.
`-central-status`	Indicates that the file status information will be stored at a central location for this user on this workstation. This location can be set using the Windows or Cross-Platform client, or this command (see the `-central-repository` option). If you do not set this location it defaults to: For Windows NT, the folder in which the application has been installed. For Windows XP: `C:\Documents and Settings\username\Local Settings\Application Data\Borland\StarTeam\` For non-Windows platforms: `/user_home_directory/.starteam-client/`
`-per-folder-status`	Indicates that each working folder will contain file status information for its own files. The information is stored in the .sbas folder, a child folder of the working folder. If you move the working folder, the file status information goes with it.
`-list`	Displays a list of the personal options as currently set in the `starteam-client-options.xml` that stores them.

Example

The following example uses `stcmd set personal-options` to indicate that file status information will be stored in a central location: `C:\JMarsh\statusinfo`.

```
stcmd set-personal-options -central-status -central-repository "C:\JMarsh\statusinfo"
```

Related Reference

Client Command-line Operations

Update File Status: stcmd update-status

When you update the status of a file, StarTeam compares the working file with the revision you checked out and the tip revision. For example, your File list may say that the file is Current, but someone else has just checked in a copy of it, so the status of your file is actually is Out Of Date.

Updating file statuses is not the same as updating files. If a file is not in your working folder, updating the status lets you know that the file's status is Missing, but will not check out the file for you. Normally, you update file status to determine whether a file should be checked in, checked out, added, or ignored.

For example, you may want to:

- Check in a file if its status is Out Of Date, Missing, or Merge.
- Check out a file if its status is Modified or Merge.
- Add a file to the application if its status is Not In View. However, the `update-status` command never lists files that have the status Not In View because they are not stored in the repository.

Use `stcmd update-status` to display the filename, its status before the command, and its status after the command. A sample line of output might be: `x.cpp: status is Current (was Unknown)`.

Syntax

The syntax for this command is as follows:

```
stcmd update-status -p "projectSpecifier" [-pwdfile "filePath"] [-cmp] [-csf]
[-encrypt encryptionType] [-is] [-q] [-x] [-stop] [-rp "folderPath" | -fp "folderPath"]
[-cfgl "labelName" | -cfgp "stateName" | -cfgd "asOfDate"] [-filter "fileStatus"] [-eol
[on | off ]][-v] [files...]
```

Option	Description
`-cfgd`	Configures the view as of the specified date/time. Examples include: "12/29/01 10:52 AM" "December 29, 2001 10:52:00 AM PST" "Monday, December 29, 2001 10:52:00 AM PST"
`-cfgl`	Configures the view using the specified label. Without `-cfgl`, `-cfgp`, or `-cfgd`, the view's current configuration is used.
`-cfgp`	Configures the view using the specified promotion state.
`-eol`	When on, calculates status of text file without using end-of-line (eol) markers. When off, which is the default, the update status is computed on working file and tip revisions as they are with their current end-of-line markers. With this option on, a working file that has LF eol markers can be compared to a tip revision with CR/LF eol markers and be considered Current if the only difference is the eol markers.
`-contents`	Sends file contents rather than MD5 checksum.
`-filter`	Specifies a string of one or more characters, each of which represents a file status. Never include spaces or other whitespace in this string. Only files that currently have the specified statuses will be updated. `C` for Current, `M` for Modified, `G` for Merge, `O` for Out of Date, `I` for Missing, and `U` for Unknown.
`-v`	Reports the status of every file in the specified folder's working folder—unless its status is Not In View. Without this option, a file's status is displayed only if it has changed.

Example

The following example uses `stcmd update-status` to verify that each file in the working folder for the StarTeam folder named `SourceCode` has an accurate status. `SourceCode` is a child of the root folder `StarDraw` (in the StarDraw view of the StarDraw project).

```
stcmd update-status -p "JMarsh:password@Orion:1024/StarDraw/StarDraw/SourceCode" "*"
```

Related Reference

Client Command-line Operations

VCM Command-line Utility

The View Compare/Merge Command-line Utility (`VCMUtility`) compares a source StarTeam view to a target view, and optionally merges the differences into the target view.

This section includes all the help topics associated with the `VCMUtility`.

View Compare/Merge is only available in the StarTeam Cross-Platform Client, and the VCMUtility.

In This Section

Overview of the VCM Command-line Utility (VCMUtility)
Explains the View Compare/Merge command-line utility called `VCMUtility`.

VCMUtility Commands
Explains the View Compare/Merge command-line utility (`VCMUtility`) command types.

VCMUtility Connection Options
Explains the View Compare/Merge command-line utility (`VCMUtility`) connection options.

VCMUtility Session Options
This section describes the `VCMUtility` options that can be used in new sessions.

VCMUtility Miscellaneous Options
Explains the View Compare/Merge (`VCMUtility`) command-line utility miscellaneous options.

VCMUtility Examples
Shows examples of using the (`VCMUtility`).

Cheat Sheet
Lists the `VCMUtility` commands and options and their syntax.

Syntax for VCMUtility Compound Options
Explains the details of the syntax requirements for each `VCMUtility` option.

Overview of the VCM Command-line Utility (VCMUtility)

The VCMUtility is a command-line utility that compares a StarTeam source view to a target view, and optionally merges the differences into the target view.

The VCMUtility documentation includes the following:

- Commands
- Connection Options
- Session Options
- Miscellaneous Options not saved in sessions

Note: You can start a View Compare/Merge session from the command line and finish it in the StarTeam Cross-Platform Client in the View Compare/Merge UI. For example, you can use the VCMUtility to create a VCM session, perhaps using its DefaultAction option, but do not let it commit. It will automatically save the VCM session with any alternate name you choose if needed. You can then open that VCM session in the StarTeam Cross-Platform Client, review and make adjustments, then commit the changes to the repository.

Syntax Conventions

The syntax for the command-line uses the following conventions.

Convention	Description
Curly braces {}	Encloses required syntax elements.
Square brackets []	Encloses optional elements
Angle brackets < >	Encloses a word or phrase that must be replaced with an appropriate value, or set of values. For example, <file name> would be replaced by an actual filename or path, and <userid> would be replaced by an actual user ID. However, many of the words or phrases in angle brackets can be expanded into more complicated syntax. For example, <change requests> can be replaced by CR, CRs, ChangeRequest, ChangeRequests, ChangeRequest *4277, and so on. When you are not sure what can be used, see the VCMUtility "Cheat Sheet" topic in the Reference/CompareMerge section of the documentation. The "Cheat Sheet" provides the full syntax for phrases like <change requests>.
Vertical bar |	Separates alternate elements.
Prefixed with an asterisk *	Indicates the following element can be repeated.

Note: All options are case-insensitive (for example, Server is the same as server).

VCMUtility Command

```
VCMUtility [<options file>] [options]
```

You can provide options in the specified <options file> (as the first parameter), command-line arguments, or both. Command-line arguments override any options found in the <options file>. In the <options file>, the option name should begin as the first character on a new line and exclude the leading -.

VCMUtility Options File

You can specify `VCMUtility` options in an options file whose name is passed as the first parameter of the `VCMUtility` command.

Example:

```
VCMUtility c:\VCMconfig.txt
```

Each option in the file must begin on a new line. Option names must begin in column 1 and be followed by at least one white space character. An option's value can flow onto multiple lines by starting each continuation line with a blank or tab character. Blank lines are ignored. You provide comments by prefixing them with a double forward-slash (`//`).

Example:

```
// This is a comment
server jsmith:mypw@somehost:49201
type Rebase
include "/Cygnus/StarTeam/<StarTeam Core>/Server/Common/*.h" +ALL
  *.cpp    *.rc Makefile    // long value continued on a second line

// The line above was blank
save
 my—rebase-session               // value provided on a separate line
```

Command-line Parameters

`VCMUtility` options can be passed as command-line parameters by placing a dash in front of the option name. For example, the `Server` option can be provided as a command-line parameter `-server`. If an option has secondary "value" tokens, they must immediately follow the option name (without a dash).

Mixing Input Sources

`VCMUtility` options can be provided in an options file, with command-line parameters, or with a mixture of both. For example, commonly-used or "static" option values can be placed in the configuration file while "dynamic" values can be provided in command-line parameters.

A command-line parameter may specify the same option as defined in the configuration file. When a command-line argument specifies the same option as in the options file, the command-line option value overrides the configuration file option value. For example, if the configuration file specifies `Source View1` but the command-line specifies `-Source View2`, then `View2` is used as the source view.

Option Values with Unicode Characters

The encoding of option values passed as command-line arguments is controlled by the launching environment (for instance, command shell). Consequently, on systems where option values must be passed to the `VCMUtility` that require characters not expressible by the launching environment, those options must be passed by way of the options file.

When the options file does not begin with a byte-order mark (BOM), it is opened with the system default character set (for example, ANSI [Windows-1252] on Windows, UTF-8 on Linux). If the options file begins with a BOM, it is interpreted with the corresponding encoding. UTF-8 and UTF-16 encodings allow the full set of Unicode characters to be provided in the options file.

For Reference, the BOM sequences are:

BOM	Encoding
0xEFBBBF	UTF-8
0xFEFF	UTF-16 BE (big-endian)
0xFFFE	UTF-16 LE (little-endian)

Boolean Options

The default for all Boolean options (whose value can be `True` or `False`) is `False`. However, specifying a Boolean option without an option value is the equivalent to specifying the value `True`. Thus, a Boolean option can be enabled by simply including it. Example:

```
// Set these options to True
AutoLogon
BreakLocks
```

Abbreviations

In addition to their "long form" (shown in this document), most command and option names have one or more "short forms" or abbreviations. These alternate spellings help shorten `VCMUtility` command tails with lots of options. The full lists of abbreviations can be achieved by using the command `-Help abbreviations`. Example abbreviations are:

`Help`: `H` or `?`.

`ActiveProcessItem`: `ActivePI` or `API`.

`SourceLabel`: `SrcLabel` or `SL`.

In most cases, a syntactic item spelled with mixed-case in this document can be abbreviated to its "capitals only" short form. For example, `ManualMergeFiles` can be `MMF`, or `AutoMergeProperties` can be `AMP`, and so on.

Exit Codes

The VCM utility will return the following exit codes to indicate the results of its execution:

Exit Code	Description
0	No errors occurred.
1	A fatal error occurred.
2	Partial success. This result is returned when the compare phase was performed, but the commit could not be performed due to unresolvable conflicts.

VCMUtility Log Files

During its execution, the `VCMUtility` writes informational, warning, and error messages to the console window (standard out). For most operations, the `VCMUtility` also creates a log file that summarizes its operation. As with console window output, the log file is more detailed when the `Verbose` option is enabled. The log file is created for new VCM sessions and for the `Import`, `Open`, `Replay`, and `Resume` commands. However, the log file is not started

unless command-line parameters and the options file, if used, have been parsed without errors. A log file is not created for the `Help` or `Delete` commands.

The VCMUtility log file is created in the user's home directory (what Java identifies as *user.home*) with the following file name:

```
VCMUtility-YYYY-MM-DD_hh-mm-ss.log
```

`YYYY-MM-DD` and `hh-mm-ss` are the current date and time in the local time zone. The full path name of the log file is written to the console window when the log file is started.

VCMUtility Support for Change Packages

Beginning with the StarTeam 2009 release, the `VCMUtility` supports change packages for any StarTeam configuration that has been upgraded to the 2009 release. Because change packages are persistent objects stored at the server, they offer many advantages and over VCM session (`.vcms`) and VCM export (`.vcmx`) files. Therefore, for StarTeam configurations that have been upgraded, change packages are preferred over session and export files for saving and resuming sessions. Correspondingly, the `Save` option without a parameter and the `Open` command are preferred over the `Save` option with a parameter, the `Resume` command, the `Export` command, and the `Import` command. However, for backward compatibility, the 2009 `VCMUtility` still supports commands that use VCM session files. See the `Open` command and the `Save` option for more information.

Related Reference

- VCMUtility Commands
- VCMUtility Connection Options
- VCMUtility Session Options
- VCMUtility Miscellaneous Options
- VCMUtility Examples
- Cheat Sheet

VCMUtility Commands

This section defines VCMUtility functionality in terms of its utility execution commands. Each VCMUtility execution performs one command.

For a description of syntax conventions, see "Overview of the VCM Command-line Utility" in the Related Reference links below.

VCMUtility Command

```
VCMUtility [<options file>] [options]
```

You can provide options in the specified <options file> (as the first parameter), command-line arguments, or both. Command-line arguments override any options found in the <options file>. In the <options file>, start option names in column 1 and exclude the leading "-"

VCMUtility Command Types

This section contains the VCMUtility command types. The default command type is a new VCM session.

New Session Command

By default, each VCMUtility execution begins a new VCM session unless the Help, OPEN, Replay, Resume, Delete, or Import command is explicitly given.

Help Command

```
?
```

```
Help [<option>]
```

Displays the VCMUtility Help. If you provide an <option>, help specific to that topic is displayed. For example, Help MMF would provide help on the ManualMergeFiles option.

Delete Command

```
Delete <VCM session file>
```

Specifies that the session stored in the specific <VCM session file> is to be deleted. All intermediate files (for example, merged result files) and the session file itself are deleted. However, if the session was previously saved as an uncommitted change package in the target view, the change package object is not deleted.

Import Command

```
Import <VCM exchange file>
```

The Import command is identical to the Resume command except that the <VCM exchange file> passed to it must be a VCM exchange file (.vcmx) previously created by an Export command. The imported VCM session is resumed where it left off:

- The compare phase is performed if it has not yet successfully completed.
- Manual merging is performed if ManualMergeFiles is specified and existing file merge conflicts exist.
- The target view "merge preview" is checked out if CheckoutPreview is specified and commit has not yet been performed.

- The differences report is generated if ReportDiffs is specified and commit has not yet been performed.
- The commit phase is performed if CommitMerge is True and commit has not yet been performed.
- The update report is generated if ReportUpdates is specified and commit has been performed.

Export and Import can be used together to "transport" a VCM session from one workstation to another. For example, one user could create a new VCM session, resolve all conflicts, then Export the session. The resulting archive file could then be transferred to a test machine, where the Import command can be used with the CheckoutPreview option (with CommitMerge set to **False**) to check out, build, and test the target merge "preview". If tests succeed, the test machine could then execute a Resume command and set CommitMerge to True.

Note: Sessions resumed by way of the Resume or Import command are automatically saved if they are not committed. If the Save option is specified, the session is saved in the specified <VCM session file>. Otherwise, the VCM session file specified by a Resume command is used; an automatically-generated VCM session filename is used for an Import command.

Open Command

Open <Change Package name>

Resumes a VCM session previously saved as a change package with the given name. This option is only available on servers that support change packages. The specified name must be the default or user-specified name of a saved, uncommitted change package belonging to the specified Project and TargetView, which are required. Also, the session must not be locked by another user, which typically indicates that it has already been opened by that user.

For additional information, see the Name, Save, Import, and Resume commands.

Replay Command

Replay <Change Package name>

Creates a new VCM session by "replaying" a previously-committed change package to a new target view. This command is only available when the server supports change packages. The named Change Package must belong to the project specified by the Project option and the view identified by the SourceView option. (Since committed change packages "belong" to the target view they update, the target view of the change package to be replayed is always the source view for the new session.)

When the Replay command is used, the TargetView should be specified, allowing the MergeType of the new session to be chosen automatically based on the relationship between the two views:

- If the target view is a child of the source view, a Rebase session is performed.
- If the target view is the parent of the source view, a Promote session is performed.
- Otherwise, a Replicate session is performed.

Alternatively, you can specify a MergeType of Promote, in which case the target view is not needed.

A replay VCM session attempts to make the same changes in the new target view that were made in the specified change package. This means that the source scope of the new VCM session is automatically chosen. Consequently, the Include and Exclude options are not allowed. In a replay session, some changes made in the original change package might not be possible in the new target view (such as when a new version is already present). Some changes may need to be applied in a different way (for example, Move-and-Merge instead of Merge), and new conflicts could appear (such as Merge instead of Repin). The replay session can be committed only if no unresolved conflicts occur.

Resume Command

`Resume <VCM session file>`

Specifies that the session saved in the given `<VCM session file>` is to be resumed instead of creating a new session. This is typically used to perform the commit phase of a previous session for which only the compare phase was performed. A session that has already been committed can also be resumed, but only to generate a difference report. For more information, see the `Export` option and the `Import` command.

Related Reference

Overview of the VCM Command-line Utility (VCMUtility)
VCMUtility Connection Options
VCMUtility Session Options
VCMUtility Miscellaneous Options
VCMUtility Examples
Cheat Sheet

VCMUtility Connection Options

This section defines VCMUtility functionality in terms of its connection options.

AutoLogon

AL

AutoLogon [True] | [False]

If a <user> is not specified in the Server option, AutoLogon requests an attempt be made to log on using the userid/password for the specified StarTeam server, as stored by the StarTeamToolbar Utility.

Encryption

Encrypt

En

Encryption {NONE | RC4 | RC2_ECB | RC2_CBC | RC2_CFB}

Specifies the encryption level of the server connection. The default is NONE. However, due to SDK behavior, if necessary, the VCMUtility will automatically upgrade the encryption level to the minimum value required by the StarTeam server.

PwdFile

PF

PwdFile <file name>

Specifies a file that contains the logon password. -PwdFile overrides the <password> if provided in the Server parameter.

Server

S

Server [<user>[:<password>]@]<host>[:<port>]

Specifies the StarTeam server to which the VCM Utility will connect.

- If <user> and AutoLogon are not specified, the logon <user> defaults to "Administrator."
- If <password> and PwdFile are not specified, the VCMUtility prompts for the password.
- If a <user> or <password> contains the characters ":" or "@", or a blank, it must be enclosed in single or double quotes.
- If a <user> or <password> is quoted, it can contain an embedded quote of the same type by escaping (preceding) it with a backslash (\).
- If a quoted <user> or <password> contains an embedded backslash, it must be escaped with another backslash. For example, a double backslash within a quoted token is interpreted as a single backslash.
- The server <host> can be a host name or IP address. The <host> is required if the Server option is specified.
- If the Server option is not specified, the <host> defaults to **localhost**. If not specified, the <port> defaults to 49201.

UseCA

`UCA`

`UseCA {<host>:<port> | AutoLocate}`

Specifies that file check-outs should attempt to use an MPX Cache Agent. The Cache Agent can be explicitly provided with a host name or address (`<host>` and port number (`<port>`), or the network-nearest Cache Agent can be automatically located (`AutoLocate`).

UseServerProfile

`USP`

`UseServerProfile [True | False]`

If true, specifies that the `<host>` name specified in the Server option should be interpreted as a server profile name. Server profiles are stored in the user's `starteam-servers.xml` file. A server profile specifies a StarTeam server host name, port number, encryption level, and compression setting. Consequently, when `UseServerProfile` is specified, the Server option must be specified but should not contain a port number, and the `Encryption` option should not be specified.

Related Reference

Overview of the VCM Command-line Utility (VCMUtility)
VCMUtility Commands
VCMUtility Session Options
VCMUtility Miscellaneous Options
VCMUtility Examples
Cheat Sheet

VCMUtility Session Options

This section describes the `VCMUtility` options that can be used in new sessions..

Session Options are grouped into two sections: "New Session Options" and "Resumed Session Options".

New Session Options

This section contains the `VCMUtility` session options.

AutoMergeFiles

`AMF`

`AutoMergeFiles [True | False]`

If `True`, requests automatic merging of files found in the compare phase in a merge state. When an auto-merge is successful, the result file is retained as part of the VCM session. Otherwise, the result file is discarded and the affected files remain in an unresolved merge state. `AutoMergeFiles` is ignored for Compare sessions.

AutoMergeProperties

`AMP`

`AutoMergeProperties [True | False]`

If `True`, requests automatic merging of properties for items found in the compare phase in a merge state. If property auto-merging is successful, the merged item is retained as part of the VCM session. Otherwise, the merged item is discarded and the items are flagged as being in an unresolved property merge state. `AutoMergeProperties` is ignored for Compare sessions.

BreakLocks

`BL`

`BreakLocks [True | False]`

If `True`, requests that an attempt is made to break any lock found on items used in the compare phase. Breaking a source or target item lock is only required when the lock is owned by another user. Lock breaking requires a special permission and may not be successful. `BreakLocks` is ignored for Compare sessions.

CaseSensitiveFilenames

`CSF`

`CaseSensitiveFilenames [True | False]`

If `True`, considers file names different only by case as unequal for purposes of evaluating the `PreventDuplicateFilenames` option and for matching files between source and target views.

CheckoutPreview

`Checkout`

`CP`

`CheckoutPreview <files> [<check-out options>]`

This option specifies that files within a "merge preview" are to be checked out to the client workspace. A "merge preview" is a simulation of the target view updated with all changes in the VCM session. The `<files>` syntax allows

file names and/or patterns to be checked out from specified folders in the merge preview. The optional <check-out options> control options such as where files are to be checked-out and what status of files should be checked-out.

When CheckoutPreview is specified, files are checked out after the compare phase, after auto- and manual-merging has occurred, but before a commit occurs. The check-out occurs only if the VCM session has no file content merge conflicts. If merge conflicts exist, an error is displayed, and no merge is performed, regardless of the CommitMerge option. If no merge conflicts exist and CommitMerge is True, the VCM session is committed after the check out is performed.

Example:

```
CheckoutPreview /src/com/acme/*.java +cwf +eol LF +filter CGMIOU +o +ro
   +rp C:\BuildDir
```

CommitMerge

Commit

CM

CommitMerge [True | False]

Specifies whether or not the results of VCM session should be committed. False specifies that a commit will not be performed. This option can be used to produce a compare/report-only session. True specifies that the commit should occur only if there are no unresolved conflicts. CommitMerge is ignored for Compare sessions.

DefaultAction

DA

DefaultAction [MergeType <merge type>] [ItemType <item type>] <match state> <action>

Specifies a default <action> for items that are compared and meet the conditions specified in the given <match state>. The VCM utility uses a rules-based "decision table" to determine what action, if any, should be taken when it finds item differences between the source and target views. The DefaultAction option allows the default rules to be overridden. This option can be specified multiple times to change the default action for multiple differences. However, the order of definition is important: if two overrides are both applicable to an item difference found in the compare phase, the last override specified takes precedence over the prior one.

- If MergeType is specified, the DefaultAction only applies to VCM sessions of the specified <merge type>: Rebase, Promote, or Replicate.
- If MergeType is not specified, the DefaultAction applies to the current VCM session.

Specifying a DefaultAction with a different <merge type> than that of the current session allows rules used by different VCM sessions to be specified in a single options file.

If ItemType is specified, the DefaultAction applies only to items of the specified <item type>: CRs, Files, Folders, Requirements, Tasks, or Topics. By default, a DefaultAction applies to items of all types.

The <match state> determines the conditions that must be met by the source and/or target items during comparison. A <match state> consists of one or more source/target <item condition> definitions, each of which has a <condition name> (for example,source.moved), and a <condition value> (True, False, or Unspecified). The <condition value> is optional and defaults to True. A <match state> is the union of all the conditions defined for it.

The <action> determines how to handle source/target item pairs whose differences match the <match state>. The <action> merely defines the default action for matching items; the actual action can be changed after compare in the StarTeam Cross-Platform Client.

Some example DefaultAction definitions are shown below:

```
//When a source item has moved, but the target item has not,
//ignore the move.
DefaultAction source.moved target.moved false Ignore

//In a Rebase, if a file is binary and has been modified in both the
//source and target, overwrite the target with the source version.
DefaultAction MergeType Rebase
   items.binaryfile
   source.modified
   target.modified
   Overwrite

//In a Promote, if a CR has moved in both the source and target views
//(to different folders), move the target item to the matching folder as
//the source item, but only if the CRs are on the same branch.
DefaultAction MergeType Promote ItemType CR
   source.moved
   target.moved
   items.branched false
   Move
```

DefaultAction is ignored for Compare sessions.

DefaultComment

DC

DefaultComment <comment>

Specifies the default revision comment to be used for new item revisions created in the target view. The <comment> is a free-form text string. Within the comment value, all white space sequences, including line breaks (CRs and LFs), blanks, and tabs, are converted into a single blank for each occurrence. By default, an auto-generated comment is used as the default revision comment for new item revisions. To disable the use of a default revision comment, specify the DefaultComment option with an empty value.

DefaultComment is ignored for Compare sessions.

Exclude

Exc

Exclude <folders>

Exclude the specified folders from the source scope. Only folders explicitly specified in Include <files> or Include <folders> are excluded. Consequently, an Exclude <folders> option can be used to "prune" unwanted folders from the source scope.

For Example:

```
//Include CRs and files in all folders below /a/b/
Include /a/b/ +all CRs Files

//But exclude CRs in folder /a/b/c/
Exclude /a/b/c/ CRs
```

```
//But if this CR is in folder /a/b/c/, it is still included
Include CR 12345
```

Regardless of declaration order, `Exclude` options are processed after `Include` options.

Include and Exclude Semantics

If no `Include` options are specified, the default VCM session scope is implicitly "all files in the source view". This is equivalent to explicitly specifying `include /* +all`. If at least one `Include` option is specified, the scope is explicitly limited to those items selected by `Include` statements. In both implicit and explicit scopes, all selected source items are pruned by any `Exclude` options.

All `Include` and `Exclude` options must identify objects (labels, files, CRs, and so on) in the source view. Also, selection type names can be singular or plural (`RevLabel`, `CR`, and so on), even if multiple values are provided.

Note: `Exclude` options are always processed after `Include` options, regardless of declaration order. Therefore, `Exclude /src/foo/bar/` followed by `Include /src/foo/ +all` causes folder `/src/foo/bar/` to be excluded.

Export

`Exp`

`Export <VCM exchange file>`

The `Export` option specifies that all the VCM session information, including merged result files, are to be combined and stored in the given `<VCM exchange file>`. The exchange file name is always suffixed with a `.vcmx` extension. A VCM exchange file allows the entire VCM session to be transported to another machine, allowing that machine to perform an `Import` command, which resumes the session. (See the `Import` command for more information.)

If the `<VCM exchange file>` does not contain path information, it is saved in the user's home directory (what Java identifies as `user.home`).

Note: The `Export` option always causes the VCM exchange file to be created, even when the session itself is not saved. See the `Save` option for more information.

FixFloatingChildShares

`FFCS`

`FixFloatingChildShares [True | False]`

Specifies whether, in Rebase and Replicate merge operations, each target view item found that is a floating share of a source view item should be “fixed” by pinning it. When a target view item is a floating child share of a source item (which implies that the target item has not branched), differences will not be detected between the source and target item during VCM sessions because changes to the source item immediately float to the child item. VCM best practices suggest that child shares should always be pinned, allowing changes to propagate from the source to target view in a controlled manner. This option allows floating child items found by VCM to be “fixed” by pinning them to the parent item revision. Specifying this option has a performance cost due to the extra commands required to check each target item examined during the compare phase.

IgnoreMergePoints

`IMP`

`IgnoreMergePoints [True] | [False]`

Specifies whether merge points should be ignored during the comparison phase. If `True`, items with merge conflicts use their branch point as the common ancestor instead of the source revision of the last merge point.

Include

`Inc`

`Include {<change requests> | <files> | <folders> | <process items> | <requirements> | <revision labels | <tasks> | <topics> }`

Includes the specified items in the source scope. The `Include` option can be provided multiple times, causing all selected items to be included. Only one item selection type (revision labels, change requests, and so on) can be specified with each `Include` option. The selection type keyword, which is optional for files and folders, can be singular or plural, for example, `ProcessItem` or `ProcessItems`.

Examples:

```
Include CRs ALL
Include /src/com/*.java +all *.jar +2 *.jpx Buildnumber.h
Include Folders /docs/api/ +all
Include ProcessItem CR 451
Include Reqs 4515 4516
Include RevLabel "Beta Fix 12.413"
Include Topic 14512
Include Task 413
```

LockMergeConflicts

`LMC`

`LockMergeConflicts {None | Source | Target | Both}`

Specifies that items with unresolved conflicts are to be locked exclusively in either the Source, Target, or Both views. Locks are acquired in the compare phase. `None` is the default, which specifies that no locks are to be created for items with unresolved locks. Note that locks are only applied to source and/or target items for which differences are found. Locks are not applied to items that are compared for which no differences are found. Also, note that this option is not affected by the Project option **Require exclusive comment when files are checked in** nor the client workstation option **Exclusively lock files on check-out**. Those options are properly handled by the VCM engine. `LockMergeConflicts` is ignored for Compare sessions.

ManualMergeFiles

`MMF`

`ManualMergeFiles [True | False]`

If `True`, this causes the file merge tool configured for the workstation to be launched for each source/target file pair found in a content merge state.

The `ManualMergeFiles` option can be used in conjunction with `AutoMergeFiles`:

- If a merge conflict is detected and `AutoMergeFiles` is requested, an auto-merge attempt is made first.
- If the conflict is resolved, the merged result file is saved, and a manual merge is not needed.
- If the auto-merge is not successful, or if `AutoMergeFiles` has not been requested, then if `ManualMergeFiles` is `True`, a manual file merge is performed.

Note: `ManualMergeFiles` is ignored (and a warning is displayed) if the workstation has no manual merge tool configured. Also, if the manual merge tool cannot be launched, or returns an error condition, the affected file remains in an unresolved conflict state. `ManualMergeFiles` is ignored for Compare sessions.

Match

`Match [Folder] *{<folder path> to <folder path>}`

Specifies that for comparison purposes, the folder specified in the first `<folder path>`, which must reside in the source view, should match the second `<folder path>`, which must reside in the target view. The `Match` option is sometimes needed to prevent “ambiguous match” conditions, which can occur when one of the views is a non-derived view. Typically, the `Match` option is only needed to match the source and target view root folders. However, other folders can be matched to resolve other ambiguous match conditions reported by the compare phase.

Both the source and target `<folder path>` must begin and end with a forward slash (`"/"`).

By convention, the root folder is represented by a single `"/"`. This means that the root folder name should not be provided in folder paths. For example, if the root folder is named “StarDraw”, the folder path for the immediate child folder “Source Code” is simply `"/Source Code/"`.

Examples:

```
// Force the source and target root view folders to match.
Match / to /
```

```
//Force the source view folder "/Source Code" to match the target view
//folder "/Modules/Materials/src".
Match "/Source Code/" to "/Modules/Materials/src/"
```

MergeType

`Type`

`MT`

`MergeType {Compare | Rebase | Promote | Replicate}`

Specifies whether to perform a Compare session or a Rebase, Promote, or Replicate merge session. If only a `SourceView` is specified, `MergeType` defaults to Promote. If only a `TargetView` is specified, `MergeType` defaults to Rebase. If both `SourceView` and `TargetView` are specified, `MergeType` must be specified. For a Compare session, the source and target views can be the same.

Name

`Na`

`Name <Change Package name>`

Specifies the name of the change package associated with the VCM session. For servers that support change packages, a name is automatically chosen when a change package is created by saving or committing the session. This option allows a specific name to be used instead of the default name. However, the name must be unique from all other change package names already saved or committed for the target view, otherwise the save or commit action will fail.

When the `Name` option is used in conjunction with the `Open` command, the opened change package is renamed to the given value.

Also see the Save and CommitMerge options.

PostCommitLabel

PostCL

PostCommitLabel <label>

If the VCM session is committed, the given *view* <label> is created in the target view after all updates are performed. The label reflects the revisions of all target view items used during the compare phase, *modified* by the changes made by the commit phase. This means the label contains new items, new item revisions, and item moves, but items deleted by the commit will be detached from the label. The post-commit label is essentially identical to the "pre-merge view". PostCommitLabel is ignored for Compare sessions.

By default, a post-commit view label is created with a default name. To disable the post-commit view label, specify PostCommitLabel with a blank value (that is, " ").

PostCommitRevLabel

PostRL

PostCommitRevLabel <label>

If the VCM session is committed, the given *revision* <label> is created in the target view, and all items modified by the VCM session, except for deleted items, are attached to it. Consequently, the label contains items that were added, moved, re-pinned, or updated in any other way (except for deletion) by the VCM session. PostCommitRevLabel is ignored for Compare sessions.

By default, a post-commit revision label is not created.

PreCommitLabel

PreCL

PreCommitLabel <label>

The given view <label> is created in the target view, reflecting the snapshot used in the compare phase. The label reflects the revisions of all target items used during the compare phase. PreCommitLabel is ignored for Compare sessions.

By default, a pre-comimt view label is not created.

PreCommitRevLabel

PreRL

PreCommitRevLabel <label>

If the VCM session is committed, the given revision <label> is created in the target view, and all non-ignored target view items are attached to it in their “before” state. That is, target view items to be modified by the session are attached to the revision label before they are modified. This means that items to be added (for example, shared) to the target view will **not** be attached, but items to be deleted **will** be attached. PreCommitRevLabel is ignored for Compare sessions.

By default, a pre-commit revision label is not created.

PreventDuplicateFilenames

PDF

`PreventDuplicateFilenames [True | False]`

If `True`, it specifies that sharing a new file to the target view is not allowed if it results in two identically-named files to exist in the same folder.

Project

`Pro`

`Project <project>`

Specifies the project to be used in the VCM session. This option is required. The source and target views must belong to the same `<project>`. Project names are case-insensitive.

ReportDiffs

`RD`

`ReportDiffs [True | False]`

If `True`, causes a report to be generated listing item differences found in the compare phase. The difference report is generated in the user's home directory (what Java identifies as `user.home`) with the following title:

```
VCMDiffReport-YYYY-MM-DD_hh-mm-ss.html
```

where `YYYY-MM-DD` and `hh-mm-ss` are the current date and time in the local time zone.

ReportUpdates

`RU`

`ReportUpdates [True | False]`

If `True`, causes a report to be generated listing all changes made to the target view in the commit phase. The update report is generated in the user's home directory (what Java identifies as `user.home`) with the following title:

```
VCMUpdateReport-YYYY-MM-DD_hh-mm-ss.html
```

where `YYYY-MM-DD` and `hh-mm-ss` are the current date and time in the local time zone.

`ReportUpdates` is ignored for Compare sessions.

Save

`Save [<VCM session file>]`

Specifies that the VCM session is to be saved. By default, uncommitted VCM sessions are automatically saved to a VCM session (`.vcms`) file with a default name using the format:

```
<user home>/VCMSession-YYYY-MM-DD_hh-mm-ss.vcms
```

where `YYYY-MM-DD_hh-mm-ss` is the date and time when the session is saved. The folder `<user home>` is the user's home directory.

If the `Save` option is specified with a `<VCM session file>` name, an uncommitted session is saved with the given file name instead of the default name. If needed, `.vcms` is appended to the name. If the given file name does not contain path information, the session file is stored in the `user.home` folder.

A `.vcms` file contains VCM session metadata, but not the contents of merged files. Merged file contents are stored in a user-relative temporary folder, referenced by elements in the session file. Consequently, a `.vcms` file can only be used to resume the VCM session on the same workstation. (See the `Resume` command.)

When the `Save` option is specified without a file name, an attempt is made to save an uncommitted VCM session as an active change package in the target view. The change package is saved with the default or user-specified name (see the `Name` option). A VCM session saved as a change package can later be resumed on any workstation using the `Open` option. However, if the server does not support change packages or a server-side save is unsuccessful, the session is instead saved to a `.vcms` file with a default file name as described above.

When a commit is successfully performed, the `Save` option is ignored. If the server supports change packages, the committed session creates a Committed change package using the default or user-specified name (see the `Name` option). If a `.vcms` file was previously created, it is deleted along with all merged result files created by the VCM session.

Also see the `Export` option.

SourceLabel

`SrcLabel`

`SL`

`SourceLabel <label>`

Requests the source view to be used as of a given view label. Label names are case-insensitive. Only one of `SourceLabel`, `SourceState`, and `SourceTime` can be specified. If none of these options is specified, the option `SourceTime Now` is implicitly used.

SourceState

`SrcState`

`SS`

`SourceState <state>`

Requests the source view to be used as of a given view promotion state. Promotion state names are case-insensitive. Only one of `SourceLabel`, `SourceState`, and `SourceTime` can be specified. If none of these options is specified, the option `SourceTime Now` is implicitly used.

SourceTime

`SrcTime`

`ST`

`SourceTime {<timestamp> | Now}`

Requests the source view to be used as of a given timestamp. The keyword `Now` causes a snapshot of the current time to be used as configuration timestamp. Only one of `SourceLabel`, `SourceState`, and `SourceTime` can be specified. If none of these options is specified, the option `SourceTime Now` is implicitly used.

SourceView

`Source`

`SV`

`SourceView <view>`

Specifies the source view to be used in the VCM session. If more than one view within the project has the same `<view>` name, a slash-separated "view path" can be provided (for example, `MainView/ChildView/GrandchildView`). If a view name contains embedded slashes, it must be enclosed in quotes.

`SourceView` is optional for Rebase merges; if specified, it must be the parent of the target view.

Note: View names are case-insensitive.

TargetLabel

`TgtLabel`

`TL`

`TargetLabel <label>`

Requests the target view to be used as of a given view label. `TargetLabel` can only be used for Compare sessions. Label names are case-insensitive. Only one of `TargetLabel`, `TargetState`, and `TargetTime` can be specified. If none of these options is specified, the option `TargetTime Now` is implicitly used.

TargetState

`TgtState`

`TS`

`TargetState <state>`

Requests the target view to be used as of a given view promotion state. `TargetState` can only be used for Compare sessions. Promotion state names are case-insensitive. Only one of `TargetLabel`, `TargetState`, and `TargetTime` can be specified. If none of these options is specified, the option `TargetTime Now` is implicitly used

TargetTime

`TgtTime`

`TT`

`TargetTime {<timestamp> | Now}`

Requests the target view to be used as of a given timestamp. `TargetTime` can only be used for Compare sessions. The keyword `Now` causes a snapshot of the current time to be used as configuration timestamp. Only one of `TargetLabel`, `TargetState`, and `TargetTime` can be specified. If none of these options is specified, the option `TargetTime Now` is implicitly used

TargetView

`Target`

`TV`

`TargetView <view>`

Specifies the target view to be used in the VCM session. If more than one view within the project has the same `<view>` name, a slash-separated "view path" can be provided (for example, `MainView/ChildView/GrandchildView`). If the view name contains embedded slashes, it must be enclosed in quotes.

`TargetView` is optional for Promote merges; if specified, it must be the parent of the source view. For Compare sessions, the target view can be the same as the source view.

Note: View names are case-insensitive.

Resumed Session Options

So that the same options file can be specified for a `Resume` command, all options allowed for new sessions can also be specified for resumed sessions. However, most options, if re-specified, are ignored because they cannot be modified once the session has been started. The only exceptions are the options specifically outlined below:

- Connection options: Since connection information (server address and port, userid, and password) are not persisted in the VCM session file, connection information must be re-specified for resumed sessions. However, a resumed session will fail if it is not reconnected to the same StarTeam server of if a different user is used that has permission conflicts with the views or items used in the VCM session.
- `CommitMerge`: This option will commonly be specified to `True` in a resumed session. This allows the original VCM utility execution to be used as a compare-only run and a second VCM utility execution to be used as a commit run.
- `ReportDiffs`: This option can be specified in a resumed session. If true, a difference report is created before the commit phase, if any.
- `ReportUpdates`: This option can be specified in a resume session. If true and the commit phase is successfully performed, all changes made to the target view are reported.
- `CheckoutPreview`: Normally, if `CheckoutPreview` was specified in the original VCM session, the “merge preview” check-out operation is performed in the resumed session with the same options as before. However, if `CheckoutPreview` is specified in the resumed session, it overrides the original option and causes files to be checked-out in the resumed session according to the new settings.
- `Description`: If specified, this option overrides the default or previously-provided Change Package description text. The new description text is used for the new Change Package revision created when the VCM session is saved or committed.
- `ManualMergeFiles`: Normally, if `ManualMergeFiles` was specified in the original VCM session, and the session is saved with unresolved file merge conflicts, the manual file merge phase will be performed again when the session is resumed. However, if `ManualMergeFiles` was not specified in the original VCM session, it can be specified as `True` in the resumed session to invoke the manual merge phase. Alternatively, it can be specified as `False` in a resumed session to prevent the manual merge phase.
- `PostCommitLabel`, `PostCommitRevLabel`, `PreCommitLabel`, and `PreCommitRevLabel`: If any of these label options are specified in a resumed option, they override the previous value for the corresponding label. When a label option is set to a blank (" "), the corresponding label option is disabled and will not be created in the commit phase.

Related Reference

Overview of the VCM Command-line Utility (VCMUtility)
VCMUtility Commands
VCMUtility Connection Options
VCMUtility Miscellaneous Options
VCMUtility Examples
Cheat Sheet

VCMUtility Miscellaneous Options

This section defines `VCMUtility` miscellaneous options that are not saved in view compare/merge sessions.

NetMon

`NM`

`NetMon [True | False]`

Enables the SDK net monitor feature. Each command issued by the `VCMUtility` to the StarTeam Server is logged to the console window (but not the `VCMUtility` log file).

Time

`T`

`Time [True | False]`

Causes timing information to be displayed for each phase of the VCM session performed. Timing information is written to both the console window and the `VCMUtility` log file.

Verbose

`vb`

`V`

`Verbose [True | False]`

Causes additional diagnostic and progress information to be displayed to the console (standard output) and to the `VCMUtility` log file during execution.

Related Reference

- Overview of the VCM Command-line Utility (VCMUtility)
- VCMUtility Commands
- VCMUtility Connection Options
- VCMUtility Session Options
- VCMUtility Examples
- Cheat Sheet

VCMUtility Examples

This topic presents examples of using the `VCMUtility` for various types of merges.

Hello World Rebase

Below are the options for the "Hello World" equivalent of a `VCMUtility` Rebase run:

```
Type     Rebase
Project     Hello
Target     World
```

Automatic Rebase

The options file below performs the same Rebase as in the previous example, but it commits if possible and provides detailed reporting on the results:

```
Type                Rebase
Project             Hello
Target              World
CommitMerge         True
LockMergeConflicts  Both

// All of these options are set to True:
AutoMergeFiles
BreakLocks
ReportDiffs
ReportUpdates
```

All files are auto-merged both in content and properties. Files that are in conflict but cannot be resolved are locked in both the source and target views. Existing lock conflicts are broken if possible. If no unresolved conflicts are encountered, the session is committed. Details of both the compare phase (differences) and commit phase (updates) are reported. If the commit is successful, all VCM session temporary files are deleted.

Promote by View Label: Compare Only

The options below perform a compare-only promote of files and CRs as of a view label, saving the session in a specific session filename:

```
// Connection settings
Server            MyUserid@ProdServer:4000
PwdFile            MyPassword.txt

// Merge type and view configuration
Type            Promote
Project            StarDraw
Source            "Beta Release"
SrcLabel        Build-4.0_142

// Select all files and CRs as source items
include            /* +all
include            / +all CRs
```

```
// Compare-only, report, and save with a specific session filename
CommitMerge        False
save            Build-4.0_142-Promote
ReportDiffs

//Miscellaneous options
AutoMergeFiles        True
AutoMergeProperties    False // leave these as conflicts and merge manually
LockMergeConflicts    Target
```

Promote by View Label: Merge

The VCM utility command-line below resumes the session saved in the previous example and commits it, assuming no new conflicts have occurred.

```
VCMUtility -resume Build-4.0_142-Promote -CommitMerge -ReportUpdates
```

Related Reference

- Overview of the VCM Command-line Utility (VCMUtility)
- VCMUtility Commands
- VCMUtility Connection Options
- VCMUtility Session Options
- VCMUtility Miscellaneous Options
- Cheat Sheet

Cheat Sheet

`VCMUtility` command-line syntax: `VCMUtility [<options file>] [*<option>]`

Within the `<options file>`, each `<option>` must begin in column 1 but can continue on subsequent lines if those lines begin with a space or tab character. When typing options in the command line, each `<option>` must be preceded with a "-".

Options

The table below lists all the `VCMUtility` command-line options and their syntax.

Command/Option

```
<option>
  <command> | <connection option> | <session option> | <miscellaneous option>
```

```
<command>
  {{Help | H | ?} [<help topic>]} |
  {Delete <VCM session file>}|
  {Import <VCM archive file} |
  {Open <Change Package name>} |
  {Replay <Change Package name>} |
  {Resume <VCM session file>}
```

```
<connection option>
  {{AutoLogon | AL} [True | False]} |
  {{Encryption | Encrypt | En} {None | RC4 | RC2_ECB | RC2_CBC | RC2_CFB}} |
  {{PwdFile  | PF} <file name>} |
  {{Server | S} [<user>[:<password>]@]<host>[:<port>]} |
  {{UseCA | UCA} {<host>:<port> | AutoLocate}} |
  {{UseServerProfile | USP} [True | False]}
```

```
<session option>
  {{AutoMergeFiles | AMF} [True | False]} |
  {{AutoMergeProperties | AMP} [True | False]} |
  {{BreakLocks | BL} [True | False]} |
  {{CaseSensitiveFilenames | CSF} [True | False]} |
  {{CheckoutPreview | check-out | CP} <files> [<check-out opotions>]} |
  {{CommitMerge | Commit | CM} [True | False]} |
  {{DefaultAction | DA} [MergeType <merge type>] [ItemType <item type>] <match state>
<action>} |
  {{DefaultComment | DC} <comment>} |
  {{Description |} <description>} |
  {{Exclude | Exc} <folders>} |
  {{Export | Exp} <VCM archive file>} |
  {{FixFloatingChildShares | True | False} |
  {{IgnoreMergePoints | IMP} [True | False]} |
  {{Include | Inc} {<change requests> | <files> | <folders> | <process items> |
     <requirements> | <revision labels> | <tasks> | <topics>}} |
  {{LockMergeConflicts | LMC} {None | Source | Target | Both}} |
  {{ManualMergeFiles | MMF} [True | False]} |
  {{Match [Folder] *{<folder path> to <folder path>}} |
  {{MergeType | Type | MT} {Compare | Rebase | Promote | Replicate}} |
  {(Name |Na} <Change Package name>} |
  {{PostCommitLabel | PostCL} <label>} |
```

```
    {{PostCommitRevLabel | PostRL} <label>} |
    {{PreCommitLabel | PreCL} <label>} |
    {{PreCommitRevLabel | PreRL} <label>} |
    {{PreventDuplicateFilenames | PDF} [True | False]} |
    {{Project | Pro} <project>} |
    {{ReportDiffs | RD} [True | False]} |
    {{ReportUpdates | RU} [True | False]} |
    {Save [<VCM session file>]} |
    {{SourceLabel | SrcLabel | SL} <label>} |
    {{SourceState | SrcState | SS} <state>} |
    {{SourceTime | SrcTime | ST} {<timestamp> | Now}} |
    {{SourceView | Source | SV} <view>} |
    {{TargetLabel | TgtLabel | TL} <label>} |
    {{TargetState | TgtState | TS} <state>} |
    {{TargetView | Target | TV} <view} |
```

```
<miscellaneous option>
  {{NetMon | NM} [True | False]} |
  {{Time | T} [True | False]} |
  {{Verbose | Vb | V} [True | False]} |
```

Other Syntax Elements

The table below lists other syntax elements in alphabetical order:

Other Syntax Elements

```
<action>
  Delete | DeleteAndReverseShare | Fail | Ignore | Merge | Move | MoveAndMerge |
MoveAndRepin | NeedsReview | Overwrite | Repin | RepinAndMove |
 ReverseShare | Share
```

```
<change requests>
  {CR | CRs | ChangeRequests} {ALL | *<CR #>}
```

```
<Change Package name>
 {A name consisting of one or more characters}
```

```
<check-out options>
  [+cwf] [+eol {on | off | cr | lf}] [+filter {CGIMOU}] [+o] [+ro]
  [+rp <work folder path>]
```

```
<condition name>
  items.binaryfile | items.branched | items.samecontent | source.childshare |
   source.deleted | source.floating | source.modified | source.moved |
   source.present | source.rootbranch | target.childshare | target.deleted |
   target.floating | target.modified | target.moved | target.present |
   target.parentdeleted | target.rootbranch
```

```
<condition value>
   True | False | Unspecified
```

```
<files>
   [File | Files] {ALL | *{<file name pattern>} [+<depth>]}}
```

```
<folder path>
{A slash followed by an optional series of folder names each ending with a slash}
```

```
<folders>
  [Folder | Folders] {ALL | *{<folder path>} [+<depth>] *[<item type>]}}
```

```
<item condition>
  <condition name> [<condition value>]
```

```
<item type>
  {ChangeRequest | CR | ChangeRequests | CRs} |
  {File | Files} |
  {Folder | Folders} |
  {Requirement | Req | Requirements | Reqs}
  {Task | Tasks}
  {Topic | Topics}
```

```
<match state>
  *<item condition>
```

```
<process items>
  ProcessItems *{[View <view>] CR <CR #> |
                 [View <view>] Req <Req #> |
                 [View <view>] Task <Task #>}
```

```
<requirements>
  {Requirement | Req | Requirements | Reqs} {ALL  | *<Req #>}
```

```
<revision labels>
  RevLabels *<label>
```

```
<task>
  {Task | Tasks} {ALL | *<Task #>}
```

```
<timestamp>
  Example formats:
    "3/11/06 1:32 PM"
    "Mar 11, 2006 1:32:38 PM"
    "March 11, 2006 1:32:38 PM PST"
    "Saturday, March 11, 2006 1:32:38 PM PST"
```

```
<topics>
  {Topic | Topics} {ALL | *<Topic #>}
```

```
<VCM exchange file>
A .vcms file name}
```

```
<VCM session file>
A .vcms file name}
```

Related Reference

- Overview of the VCM Command-line Utility (VCMUtility)
- VCMUtility Commands
- VCMUtility Connection Options
- VCMUtility Session Options
- VCMUtility Miscellaneous Options
- VCMUtility Examples
- Syntax for VCMUtility Compound Options
- <action>
- <change requests>
- <check-out options>
- <files>
- <folders>
- <process item>
- <requirements>
- <revision labels>
- <tasks>
- <timestamp>
- <topics>

Syntax for VCMUtility Compound Options

The following compound syntax items are used in the `VCMUtility` options.

In This Section

<action>

Specifies the action to perform for a given source/target item difference. An `<action>` is one of the following mnemonics:

Mnemonic	Description
`Delete`	Delete the target item.
`DeleteAndReverseShare`	Equivalent to a `Delete` followed by a `ReverseShare`.
`Fail`	Synonym for `NeedsReview` (see below).
`Ignore`	Take no action.
`MarkResolved`	Create a merge point only that marks the source and target items as resolved.
`Merge`	Merge the source and target items.
`Move`	Move the target item to the equivalent folder as the source item.
`MoveAndMerge`	Equivalent to a `Move` followed by a `Merge`.
`MoveAndOverwrite`	Equivalent to a `Move` followed by an `Overwrite`.
`MoveAndRepin`	Equivalent to a `Move` followed by a `Repin`.
`NeedsReview`	Force a review before a commit. That is, do not allow commit while this action is selected. Item differences with this action are conflicts, therefore, their action must be changed to something else.
`Overwrite`	Overwrite the target with the contents of the source.
`Repin`	Change the revision to which the target is pinned to match the source item.
`ReverseShare`	Move the source item to the target view and share it back to the source view.
`Share`	Share the source item to the target view.

Note: Not every `<action>` is valid for every item difference For example, `Delete` is not valid when the target item is already deleted.

Related Reference

Syntax for VCMUtility Compound Options
<change requests>
<check-out options>
<files>
<folders>
<match state>
<process item>
<requirements>
<revision labels>
<tasks>
<timestamp>
<topics>
Cheat Sheet

<check-out options>

The following section describes the syntax used for the compound `VCMUtility` option `<check-out options>`.

```
[+cwf] [+eol {on | off | cr | lf | crlf}] [+filter {CGIMOU}] [+o] [+ro] [+rp <work folder path>]
```

Specifies non-default check-out options. The available check-out options are similar to those provided by the StarTeam command-line (stcmd) and bulk check-out (BCO) utilities, except that option names must be prefixed with a '+' sign. The available options are detailed below.

+cwf

Requests the creation of working folders for all specified folders, even if they do not have files to be checked-out by this run. Only visible folders are created.

+eol <eol option>

Requests conversion of all end-of-line delimiters for text files to the specified format. An `<eol option>` of `on` uses the client-configured EOL format. `off` prevents any EOL conversion. `cr`, `lf`, and `crlf` cause each EOL to be converted to a carriage-return, line-feed, or carriage-return/line-feed pair, respectively. Note that text files with a "fixed" EOL format are always converted to the specified format.

+filter

```
+filter {CGIMOU}
```

Specifies the status of files to consider for check-out: **C**urrent, mer**G**e, m**I**ssing, **M**odified, **O**ut-of-date, or **U**nknown. Multiple status flags can be combined. If `+filter` is not specified, the default filter is `IO` (**Missing** and **Out-of-date**). If **Merge**, **Merge**, or **Unknown** files are included without the `+o` option, a warning is generated for each such file, and the file is not checked out.

+o

Specifies that, in addition to **Missing** and **Out-of-date** files, files whose status is **Modified**, **Merge**, or **Unknown** are included. Furthermore, all files are overwritten without warning. If `+filter` is also specified, only the specified files are checked out.

+ro

Sets each file to read-only after check out. By default, checked-out files are read-write.

+rp

Specifies the root working folder of the "merge preview". Files are checked-out to child working folders relative to `<work folder path>`

Related Reference

- Syntax for VCMUtility Compound Options
- <action>
- <change requests>
- <files>
- <folders>
- <match state>
- <process item>
- <requirements>
- <revision labels>
- <tasks>
- <timestamp>
- <topics>
- Cheat Sheet

<change requests>

`<change requests>{CR | CRs | ChangeRequests} {ALL | *<CR #>}`

Specifies all change requests in the view, or individual change requests by change request number. `CRs` and `ChangeRequests` are synonyms; the singular form of each is also accepted.

Related Reference

- Syntax for VCMUtility Compound Options
- <action>
- <check-out options>
- <files>
- <folders>
- <match state>
- <process item>
- <requirements>
- <revision labels>
- <tasks>
- <timestamp>
- <topics>
- Cheat Sheet

<files>

`[File | Files] {ALL | *{<file name pattern> [+depth>]}}`

Specifies all files in the view or a set of specific files, given as a list of file names and/or patterns, each with an optional folder `<depth>`. The keyword `File` (or `Files`) is optional unless the keyword `All` is used. A `<file name pattern>` can be a specific file name (for example, `foo.java`), a file name pattern (for example, `*.java`), or a file name or pattern with a folder path (for example, `(/src/com/acme/foo.java)` or `/src/com/acme/*.java`).

Usage

Folder paths must use forward slashes; a single slash (`/`) is a synonym for the root folder. (Consistent with other StarTeam utilities, the root folder name, which typically matches the view name, should **not** be provided in path names.)

- If a filename or pattern is provided without a folder path, the implied folder is the same as the previous `<file name pattern>` parameter.
- If the first `<file name pattern>` parameter does not contain a folder path, the root folder is implied.
- If provided, the folder `<depth>` specifies the number of child folder levels below the specified folder to include; it can be a number or the keyword **All**.
- If a file or pattern name contains spaces, it must be enclosed in quotes.

Examples

Below are examples of <files> usage:

```
// all files in the view
include Files ALL

//foo.java and bar.java in folder /src/com/acme
include /src/com/acme/foo.java bar.java

// all .java files in folder /src/com/acme and below
include /src/com/acme/*.java +all

// all .txt files in the root folder, all .zip file in first-level
// child folders, and a specific readme.txt file
include *.txt *.zip +1 /docs/acme/readme.txt
```

Related Reference

- Syntax for VCMUtility Compound Options
- <action>
- <change requests>
- <check-out options>
- <folders>
- <match state>
- <process item>
- <requirements>
- <revision labels>
- <tasks>
- <timestamp>
- <topics>
- Cheat Sheet

<folders>

`[Folder | Folders] {ALL | *{<folder path> [+<depth>] *[<item type>]}}`

Specifies all folders in the view or specific folder paths, optionally indicating a folder depth and specific item types. The keyword `Folder` or (`Folders`) is optional unless the keyword `ALL` is used.

Usage

A valid `<folder path>` must begin and end with a forward slash (`/src/com/`). If provided, the `<depth>` specifies the number of child folder levels below the specified folder to include; it can be a number, or the keyword `All`.

- If a folder path contains spaces, it must be inclosed in quotes.
- If no `<item type>` parameters are provided, only files are included in the specified folder(s). Otherwise, all items of the specified item types are included.

Recognized item types are `CRs`, `Files`, `Folders`, `Tasks`, `Topics`, and `Requirements` (singular or plural).

Examples

Below are examples of <folder> usage:

```
// all folders in the view
include folders ALL

// all files in the folder /src/com/acme/ alone
include /src/com/acme/

// all files and tasks in  /src/ and below
include /src/ +all files tasks

// all CRs in the folder "/triage/" and all files in "/PR docs/"
// child folders two levels below it
include /triage/ CRs "/PR docs/" +2
```

By convention, the root folder is represented by a single "`/`". This means that the root folder name should not be provided in folder paths. For example, if the root folder is named "StarDraw", the folder path for the immediate child folder "Source Code" is simply `/Source Code/`.

Related Reference

- Syntax for VCMUtility Compound Options
- <action>
- <change requests>
- <check-out options>
- <files>
- <match state>
- <process item>
- <requirements>
- <revision labels>
- <tasks>
- <timestamp>
- <topics>
- Cheat Sheet

<item type>

`<item type>`

Specifies an item type. Allowed values are `ChangeRequest` (or `CR`), `File`, `Folder`, `Requirement` (or `Req`), `Task`, and `Topic`. Item type names are case-insensitive and can be plural.

Related Reference

- Syntax for VCMUtility Compound Options
- <action>
- <change requests>
- <check-out options>
- <files>
- <folders>
- <process item>
- <requirements>
- <revision labels>
- <tasks>
- <timestamp>
- <topics>
- Cheat Sheet

<match state>

*<item condition>

Defines a set of conditions that apply to source/target item differences. A <match state> is the union of each <item condition> defined for it. Each <item condition> has the form:

<condition name> [<condition value>]

<condition name>

The valid <condition names> and their meaning are:

<condition name>	Meaning
items.binaryfile	Indicates whether either of the items in question is a binary file.
items.branched	Indicates whether the source and target items are in different branches of the object version tree.
items.samecontent	Indicates whether the source and target items have the same user-modifiable properties and, for files, data content.
source.childshare	Indicates whether the source item is a child share of the target item.
source.deleted	Indicates whether the item in question is deleted in the source view.
source.floating	Indicates whether the source item has a floating configuration.
source.modified	Indicates whether the item in question is modified in the source view.
source.moved	Indicates whether the item in question is moved in the source view.
source.present	Indicates whether the item in question is present in the source view.
source.rootbranch	Indicates whether the source item is the root branch of its share tree.
target.childshare	Indicates whether the target item is a child share of the source item.
target.deleted	Indicates whether the item is question is deleted in the target view.
target.floating	Indicates whether the target item has a floating configuration.
target.modified	Indicates whether the item in question is modified in the target view.
target.moved	Indicates whether the item in question is moved in the target view.
target.parentdeleted	Indicates whether the target item's folder has been deleted.
target.present	Indicates whether the item in question is present in the target view.
target.rootbranch	Indicates whether the target item is the root branch of its share tree.

<condition value>

The valid <condition value>s are:

<condition value>	Meaning
True	The condition is true for the applicable item(s).
False	The condition is false for the applicable item(s).
Unspecified	The condition is unknown or not relevant for the applicable item(s).

The `<condition value>` is optional and defaults to `True`. For any given `<match state>`, all unspecified conditions are initially `Unspecified`.

An `<item condition>` can be defined as `True` or `False` to cause the corresponding condition to "participate" in matching the condition to actual item differences.

A condition can be defined as `Unspecified`, for example, to experimentally remove the condition from the matching criteria without deleting the condition from an options file.

Note: Some conditions are mutually exclusive: if defined together, they will never match any actual item differences. For example, a source item cannot be both present (`source.present=true`) and deleted (`source.deleted=true`).

Related Reference

Syntax for VCMUtility Compound Options
<action>
<change requests>
<check-out options>
<files>
<folders>
<process item>
<requirements>
<revision labels>
<tasks>
<timestamp>
<topics>
Cheat Sheet

<process item>

```
ProcessItems *{[View <view>] CR <CR #> | [View <view>] Req <Req #> | [View <view>]
Task <Task #>}
```

Specifies a set of process items (change requests, tasks, and/or requirements) to be included. Specifying a process item causes items linked to it in the source view to be included as well. The keyword `ProcessItems` can be singular. The full names `ChangeRequest` and `Requirement` can be used in place of `CR` and `Req` respectively.

By default, a process item specified must reside in the source view. However, the optional prefix `View <view>` can be used to select a process item in a view other than the source view. When a non-source view process item is included, the process item is **not** included in the source scope, but those items linked to it in the source view are included. The specific revision of each source view item linked to the process item is included.

Examples:

```
// Include CR #451 in the source view and its linked items
include ProcessItem CR 451

//Include the items in the source view that are linked to Task #909
//includle Requirement #518, both from view "Triage"
//include ProcessItem View Triage Task 909
                    View Triage Requirement 518
```

Note: If the view name contains spaces, it must be quoted (`"Release 4.3"`). If more than one view in the project has the same view name, the view name can be a slash-separated view path (`"Apps/Releases/Release 4.3"`).

Related Reference

- Syntax for VCMUtility Compound Options
- <action>
- <change requests>
- <check-out options>
- <files>
- <folders>
- <match state>
- <requirements>
- <revision labels>
- <tasks>
- <timestamp>
- <topics>
- Cheat Sheet

<requirements>

`{Reqs | Requirements} {ALL | *{<Req #>}`

Specifies individual requirements by requirement number. `Reqs` and `Requirements` are synonyms; the singular form of each is also accepted.

Related Reference

- Syntax for VCMUtility Compound Options
- <action>
- <change requests>
- <check-out options>
- <files>
- <folders>
- <match state>
- <process item>
- <revision labels>
- <tasks>
- <timestamp>
- <topics>
- Cheat Sheet

<revision labels>

`RevLabels *<label>`

Specifies all the items attached to each specified revision label (`<label>`). The keyword `RevLabels` can be singular. Revision labels are case-insensitive.

Related Reference

- Syntax for VCMUtility Compound Options
- <action>
- <change requests>
- <check-out options>
- <files>
- <folders>
- <match state>
- <process item>
- <requirements>
- <tasks>
- <timestamp>
- <topics>
- Cheat Sheet

<tasks>

```
Tasks {ALL | *{<Task #>}
```

Includes the specified individual tasks by task number. The keyword `Tasks` can be singular.

Related Reference

- Syntax for VCMUtility Compound Options
- <action>
- <change requests>
- <check-out options>
- <files>
- <folders>
- <match state>
- <process item>
- <requirements>
- <revision labels>
- <timestamp>
- <topics>
- Cheat Sheet

<timestamp>

A `<timestamp>` must have one of the Java-recognized formats for date and time strings.

- Date formats are interpreted with the local date formatting conventions (for example, `3/11/06` is interpreted as March 11, 2006 in the United States.)
- Seconds are optional (for example, `1:32` and `1:32:00` are identical).
- The `AM/PM` indicator is required.
- The time zone indicator is optional; if omitted, the local time zone is assumed.
- The day of week, if provided, is ignored.

Examples:

```
"3/11/06 1:32 PM"
"Mar 11, 2006 1:32:38 PM"
"March 11, 2006 1:32:38 PM PST"
"Saturday, March 11, 2006 1:32:38 PM PST"
```

Related Reference

Syntax for VCMUtility Compound Options
<action>
<change requests>
<check-out options>
<files>
<folders>
<match state>
<process item>
<requirements>
<revision labels>
<tasks>
<topics>
Cheat Sheet

<topics>

```
Topics {ALL | *{<Topic #>}
```

Includes the specified individual topics by topic number. The keyword `Topics` can be singular.

Related Reference

- Syntax for VCMUtility Compound Options
- <action>
- <change requests>
- <check-out options>
- <files>
- <folders>
- <match state>
- <process item>
- <requirements>
- <revision labels>
- <tasks>
- <timestamp>
- Cheat Sheet

Index