== CMD\_DESCRIPTION\_ACL ==

Sets permissions on an object.

== CMD\_USAGE\_ACL ==

Usage:

cm ^acl (--^user=<usr\_name> | --^group=<group\_name>)

(-^allowed|-^denied|-^overrideallowed|-^overridedenied=+|-<permission>[,...])[,...]

<objectspec>

--^user User name.

--^group Group name.

-^allowed Enables the specified permission or permissions. Use a

comma to separate permissions. (Use 'cm ^showpermissions'

to display all the available permissions.)

-^denied Denies the specified permission or permission. Use a

comma to separate permissions. (Use 'cm ^showpermissions'

to display all the available permissions.)

-^overrideallowed Overrides the allowed permission or permissions. Use a

comma to separate permissions. (Use 'cm ^showpermissions'

to display all the available permissions.)

-^overridedenied Overrides the denied permission or permissions. Use a

comma to separate permissions. (Use 'cm ^showpermissions'

to display all the available permissions.)

objectspec The object whose permissions will be set.

The valid objects for this command are:

repserver, repository, branch, changeset, label, item,

and attribute.

(Use 'cm ^help ^objectspec' to learn more about specs.)

Special usage for secured paths:

cm ^acl [(--^user=<usr\_name> | --^group=<group\_name>)

(-^allowed|-^denied|-^overrideallowed|-^overridedenied=+|-<permission>[,...])[,...]]

[--^delete] [--^branches=[+ | -]<branch>[,...]]

<spec>

--^delete Removes a secured path.

See Remarks for more info.

--^branches Sets the secured path permissions to a group of branches.

Use a comma to separate branches.

Optionally, each branch can be preceded by the + or -

sign to specify whether a branch must be added or deleted

to the list when editing.

See Remarks for more info.

spec The secured path where to set the permissions.

== CMD\_HELP\_ACL ==

Configuring permissions requires understanding how Plastic SCM security works.

Check the Security Guide to learn how permissions work:

https://www.plasticscm.com/download/help/securityguide

Remarks:

This command sets permissions for a user or group on the specified objects,

repositories, branches, labels and/or server paths.

Object specs:

(Use 'cm ^help ^objectspec' to learn how to specify objects.)

The '^acl' command uses a special type of spec: secured paths.

- Secured paths specs:

^path:server\_path[#tag]

Examples: ^path:/src/foo.c

^path:/doc/pdf

^path:/doc/pdf#documents

Permission action:

Use -^allowed and -^denied to specify what permissions to set.

Use -^overrideallowed and -^overridedenied arguments to specify what

permissions to override.

Each action requires a permission list separated by commas.

Permission names:

Each permission name is preceded by + or - symbol.

The + symbol sets the permission and the - symbol clears it.

To see the permissions of an object, use the 'cm ^showacl' command.

Overridden permissions:

Overriding a permission using -^overrideallowed and -^overridedenied

allows you to bypass inheritance.

It is helpful to bypass permissions set at the repository or server

level.

Example:

cm ^acl --^user=vio -^allowed=+^ci -^overrideallowed=+^ci ^br:qa@test

(Allows user 'vio' to checkin on the branch 'qa' on repo 'test'

even if she has the permission denied at the repo level.)

Server path permissions (a.k.a. secured paths):

- It is possible to specify permissions for a given server path.

- These permissions are checked during the checkin operation.

- These permissions can also be checked during the update operation,

and can be used as a way to prevent certain directories and files to

be downloaded to the workspace.

- For every item to checkin, the server tries to match the item path

with a secured path. If it matches, the checkin operation checks

whether the item has permissions to be checked in.

The permissions that can be defined for a secured path are the

following:

'^ci', '^change', '^add', '^move', '^rm', '^read'

If the permissions check is not successful for any of the involved

items, the checkin operation will be rolled back.

To set secured path permissions to a group of branches, use the

--^branches option.

Example:

cm ^acl --^user=jo -^denied=+^ci ^path:/src#rule0 --^branches=main,main/rel0

To edit the ACL associated to the secured path, the tag is useful.

Example:

cm ^acl --^user=jo -^denied=+^rm ^path:/src#rule0

(Without the tag, the list of branches would need to be specified

again.)

The list of branches of the secured path can be edited.

Example:

cm ^acl ^path:/src#rule0 --^branches=-main,+main/rel1

(Removes 'main' from the list and adds 'main/rel1'.)

To remove a secured path, use the --^delete argument.

Example:

cm ^acl --^user=jo --^delete ^path:/src#rule0

Inheritance:

Inheritance is an option that comes from the days of Plastic SCM 3.0.

It is advanced, but almost deprecated.

It lets an object inherit its permissions from any other object,

overriding the default inheritance relationships.

Use the option -^cut to cut the inheritance chain.

Use the option -^cutncpy to cut and copy the current inherited

permissions. (This is inspired on the Windows filesystem permissions

where you can cut inheritance but retain the actual permissions.)

The -^inherit option allows the user to inherit from an object spec.

Example: '-^inherit=object\_spec'

Examples:

cm ^acl --^user=danipen -^denied=+^ci ^rep:core

(Denies checkin for user 'danipen' on repo 'core'.)

cm ^acl --^group=developers -^allowed=+^view,-^read -^denied=+^chgperm ^br:main

(The command grants view permission, clears read permission,

and denies chgperm permission to 'developers' group in 'main' branch.)

Secured path examples:

cm ^acl --^group=devs -^denied=+^ci ^path:/server#rel --^branches=main,main/2.0

(The command denies the checkin permission to 'devs' group for any path

that matches '/server' in the branches 'main' and 'main/2.0'. The tag '#rel'

is created to be able to refer to it later.)

cm ^acl ^path:/server#rel --^branches=-/main,+/main/Rel2.1

(Updates the secured path '/server', whose tag is 'rel', removing the

'main' branch and adding the branch 'main/Rel2.1' to the branch

group the secured path applies to. Considering the previous example,

now the branches list will contain 'main/Rel2.1' and 'main/2.0'.)

cm ^acl --^user=vsanchezm -^allowed=-^read -^overrideallowed=+^read ^path:/doc

(Removes '^read' permission to 'vsanchezm' overriding it in '/doc' path.)

== CMD\_DESCRIPTION\_ACTIVATEUSER ==

Activates a licensed user.

== CMD\_USAGE\_ACTIVATEUSER ==

Usage:

cm ^activateuser | ^au <user-name>[ ...] [--^server=<rep-server-spec>]

user-name The user name or user names to activate. Use double quotes (" ")

to specify user names containing spaces. Use a whitespace to

separate user names.

Options:

--^server=<rep-server-spec> Activates the user in the specified server.

If no server is specified, executes the command

in the default server in the client.conf file.

(Use 'cm ^help ^objectspec' to learn more about

repserver specs.)

== CMD\_HELP\_ACTIVATEUSER ==

Remarks:

To activate a user, it must have been previously deactivated.

By default, a user is activated the first time they perform a write

operation in Plastic SCM. The user is automatically activated only if

the maximum number of users has not been exceeded.

See the 'cm ^help ^deactivateuser' command for more information about

deactivating Plastic SCM users.

Examples:

cm ^activateuser john

cm ^activateuser david "mary collins"

cm ^au peter --^server=localhost:8087

== CMD\_DESCRIPTION\_ADD ==

Adds an item to version control.

== CMD\_USAGE\_ADD ==

Usage:

cm ^add [-^R | -^r | --^recursive] [--^silent] [--^ignorefailed]

[--^skipcontentcheck] [--^coparent] [--^filetypes=<file>] [--^noinfo]

[--^format=<str-format>] [--^errorformat=<str-format>]

<item-path>[ ...]

item-path The item or items to add. Use double quotes (" ") to specify

paths containing spaces. Use a whitespace to separate items.

Use \* to add all the contents of the current directory.

Options:

-^R -^r --^recursive Adds items recursively.

--^silent Does not show any output.

--^ignorefailed If an item cannot be added, the add operation will

continue without it. Note: If a directory cannot be

added, its content is not added.

--^skipcontentcheck When the extension is not enough to set the file as

text or binary, it will be set as binary instead of

checking the content to detect the type. This is done

to increase performance on huge checkins.

--^coparent Runs a checkout of the parent of the item being added.

--^filetypes The filetypes file to use. Check the following link for

more information:

http://blog.plasticscm.com/2008/03/custom-file-types.html

--^noinfo Doesn't print progress information.

--^format Retrieves the output message in a specific format. Check

the examples for more information.

--^errorformat Retrieves the error message (if any) in a specific

format. Check the examples for more information.

== CMD\_HELP\_ADD ==

Remarks:

Requirements to add items:

- The parent directory of the item to add must have been previously added.

Reading input from stdin:

The '^add' command can read paths from stdin. To do this, pass a single dash

"-".

Example: cm ^add -

Paths will be read until an empty line is entered.

This allows you to use pipe to specify which files to add.

Example:

dir /S /B \*.c | cm ^add -

(In Windows, adds all .c files in the workspace.)

Examples:

cm ^add file1.txt file2.txt

(Adds 'file1.txt' and 'file2.txt' items.)

cm ^add c:\workspace\file.txt

(Adds 'file.txt' item in path 'c:\workspace'.)

cm ^add -^R c:\workspace\src

(Recursively adds 'src'.)

cm ^add -^R \*

(Recursively adds all the contents of the current directory.)

cm ^add -^R \* --^filetypes=filetypes.conf

(Recursively adds all the contents of the current directory, using

'filetypes.conf' to assign a type to each file based on its extension,

instead of checking its content.)

cm ^add --^coparent c:\workspace\dir\file.txt

(Adds 'file.txt' to source control, and performs a checkout of 'dir'.)

cm ^add -^R \* --^format="ADD {0}" --^errorformat="ERR {0}"

(Recursively adds all the contents of the current directory, printing

'^ADD <item>' for successfully added files, and '^ERR <item>' for items that

could not be added.)

== CMD\_USAGE\_ADDIGNOREPATTERN ==

Usage:

cm ^addignorepattern <pattern>[ ...]

[--^workspace=<wkpath> | --^allworkspaces] [--^remove]

== CMD\_DESCRIPTION\_ADMIN ==

Executes administrative commands on the server.

== CMD\_USAGE\_ADMIN ==

Usage:

cm ^admin <command> [options]

Available commands:

^readonly

To get more information about each command run:

cm ^admin <command> --^usage

cm ^admin <command> --^help

== CMD\_HELP\_ADMIN ==

Remarks:

Only the server administrator can execute administrative commands.

Examples:

cm ^admin ^readonly ^enter

cm ^admin ^readonly ^status

== CMD\_DESCRIPTION\_ADMIN\_READONLY ==

Enables/disables the server readonly mode.

== CMD\_USAGE\_ADMIN\_READONLY ==

Usage:

cm ^admin ^readonly (^enter | ^leave | ^status) [<server>]

Actions:

^enter The server enters read-only mode. Write operations will be rejected.

^leave The server leaves read-only mode.

^status Shows the server read-only mode status.

Options:

server Executes the command in the specified server (server:port). (Use

'cm ^help ^objectspec' to learn more about server specs.)

If no server is specified, the command works with the server of the

current workspace.

If the current path is not in a workspace, the command works with

the default server defined in the client.conf config file.

== CMD\_HELP\_ADMIN\_READONLY ==

Remarks:

Only the server administrator can enter the server readonly mode.

Examples:

cm ^admin ^readonly ^enter diana:8086

cm ^admin ^readonly ^leave

== CMD\_DESCRIPTION\_ANNOTATE ==

Shows the changeset where each line of a file was last modified and its author.

== CMD\_USAGE\_ANNOTATE ==

Usage:

cm ^annotate | ^blame <spec>[ ...]

[--^format=<str\_format>]

[--^ignore=(^eol | ^whitespaces | ^"eol&whitespaces" | ^none)]

[--^dateformat=<str\_date\_format>]

[--^encoding=<name>]

[--^stats]

[--^repository=<repspec>]

spec The spec of the file to annotate.

(Use 'cm ^help ^objectspec' to learn more about specs.)

Use double quotes (" ") to specify paths containing spaces.

Options:

--^format Retrieves the output message in a specific format. See

Remarks for more info.

--^ignore Sets the specified comparison method.

See Remarks for more info.

--^dateformat Sets the output format to print dates.

--^encoding Specifies the output encoding, i.e.: utf-8.

See the MSDN documentation at

http://msdn.microsoft.com/en-us/library/system.text.encoding.aspx

to get the table of supported encodings and its format,

(at the end of the page, in the "Name" column).

--^stats Shows statistics information.

--^repository Specifies a repository spec used to calculate

the annotations. By default, this command uses the

repository where the loaded revision repository in the

workspace is stored. (Use 'cm ^help ^objectspec' to learn

more about repspecs.)

== CMD\_HELP\_ANNOTATE ==

Remarks:

Binary files can't be annotated.

--^ignore options:

^none Detects end of line and whitespace differences.

^eol Ignores end of line differences.

^whitespaces Ignores whitespace differences.

^"eol&whitespaces" Ignores end of line and whitespace differences.

--^format options:

The output parameters of this command are the following:

{^owner} User who changed the line the last time.

{^rev} Source revision specification of the line.

{^content} Line content.

{^date} Date when the line was checked in.

{^comment} Comment of the source revision of the line.

{^changeset} Changeset of the source revision of the line.

{^line} Line number of the file.

{^id} Item id.

{^parentid} Parent id of the item.

{^rep} Repository of the item.

{^branch} Branch of the source revision of the line.

{^ismergerev} Whether the revision of the line was created in a merge.

--^dateformat:

To specify the output format in which dates will be printed.

See the supported formats specified at:

https://docs.microsoft.com/en-us/dotnet/standard/base-types/custom-date-and-time-format-strings

--^repository:

To retrieve data from a remote repository. Useful for distributed

scenarios.

Examples:

cm ^blame c:\workspace\src --^ignore=^"eol&whitespaces" --^encoding=utf-8

cm ^annotate c:\workspace\file.txt --^ignore=^eol

cm ^annotate c:\workspace\file.txt --^format="{^owner} {^date, 10} {^content}"

(Writes the owner field, then a blank, then the date field (aligned to

right), then a blank, and the content.)

cm ^blame c:\workspace\file.txt --^format="{^owner, -7} {^comment} {^date}" \

--^dateformat=yyyyMMdd

(Writes the owner field in 7 spaces (aligned to the left) followed by

a blank, then the comment, followed by another blank, and ending with the

formatted date (for example, 20170329).)

cm ^annotate c:\workspace\file.txt --^repository=centralRep@myserver:8084

cm ^blame ^serverpath:/src/client/checkin/Checkin.cs#^cs:73666

(Annotates the file starting in changeset 73666 using a server path.)

== CMD\_DESCRIPTION\_APPLY\_LOCAL ==

Checks for local changes (locally moved, locally deleted, and locally changed)

and applies them, so that Plastic SCM starts tracking those changes.

== CMD\_USAGE\_APPLY\_LOCAL ==

Usage:

cm ^applylocal | ^al [--^dependencies] [<item\_path>[ ...]]

[--^machinereadable [--^startlineseparator=<sep>]

[--^endlineseparator=<sep>] [--^fieldseparator=<sep>]]

Options:

--^dependencies Adds local change dependencies into the items to

apply.

item\_path Items to be applied. Use a whitespace to separate

paths. Use double quotes (" ") to specify paths

containing spaces.

--^machinereadable Outputs the result in an easy-to-parse format.

--^startlineseparator Used with the '--^machinereadable' flag, specifies how

the lines should start.

--^endlineseparator Used with the '--^machinereadable' flag, specifies how

the lines should end.

--^fieldseparator Used with the '--^machinereadable' flag, specifies how

the fields should be separated.

== CMD\_HELP\_APPLY\_LOCAL ==

Remarks:

If --^dependencies and <item\_path> are not specified, the operation involves

all the local changes in the workspace.

It is always applied recursively from the given path.

Examples:

cm ^applylocal foo.c bar.c

cm ^applylocal .

(Applies all local changes in the current directory.)

cm ^applylocal

(Applies all local changes in the workspace.)

cm ^applylocal --^machinereadable

(Applies all local changes in the workspace, and prints the result in a

simplified, easier-to-parse format.)

cm ^applylocal --^machinereadable --^startlineseparator=">" \

--^endlineseparator="<" --^fieldseparator=","

(Applies all local changes in the workspace, and prints the result in a

simplified, easier-to-parse format, starting and ending the lines and

separating the fields with the specified strings.)

== CMD\_DESCRIPTION\_ARCHIVE ==

Archives data in external storage.

== CMD\_USAGE\_ARCHIVE ==

Usage:

cm ^archive | ^arch <revspec>[ ...] [-^c | --^comment=<str\_comment>]

[--^file=<base\_file>]

(Extracts data from the repository and stores it on external storage.)

cm ^archive | ^arch <revspec>[ ...] --^restore

(Restores previously archived revisions back into the repository.)

revspec One or more revision specs. Can be read from the STDIN

with the "-" modifier. (Use 'cm ^help ^objectspec' to

learn more about revspecs.)

--^restore Restores previously archived data from generated archive

files.

Options:

-^c | --^comment Sets a comment in the archive storage files to create.

--^file Name prefix and (optional) path for the new archive

data files.

== CMD\_HELP\_ARCHIVE ==

Remarks:

This command extracts data from the repository database and store it on

external storage, saving database space.

The command can also restore previously archived revisions back into the

repository database (--^restore).

Use 'cm ^help ^objectspec' to learn how to specify a revspec.

The user running this command must be the Plastic SCM server administrator

(repository server owner) to be allowed to complete the operation.

Every data segment from the specified revisions will be stored in a

different file, with a name starting with the value defined by the --^file

argument. This argument can contain either a full path value including a

prefix for future archive files or just this prefix value.

Once archived, the data from the specified revisions will be accessible in

two ways:

- From the client: The client will detect if the data was archived and it

will prompt the user to enter the location of the files.

Users can configure the external data location by creating a file named

externaldata.conf (at the standard configuration files locations, using

the same rules that apply for the client.conf file) containing the paths

where archived data have been located.

- From the server: This way users won't have to know whether the data was

archived or not, since requests will be transparently resolved by the

server. To do so, the administrator will create a file called

externaldata.conf in the server directory and will fill it with the

paths where the archived volumes are.

To unarchive (restore) a revision (or set of revisions), the archived

files must be accessible from the client. Hence, it is not possible to

unarchive data being resolved by the server (method 2) because the client

will not be able to identify it as archived. If method 2 is used, to

unarchive successfully, the administrator will have to edit the

externaldata.conf server file first to remove access to the archived

files which have to be unarchived.

Set PLASTICEDITOR environment variable to specify an editor to type the

comment.

Examples:

cm ^archive bigfile.zip#^br:/main

(Archives the last revision of 'bigfile.zip' in branch 'main'.)

cm ^archive ^rev:myfile.pdf#^cs:2 -^c="big pdf file" --^file=c:\arch\_files\arch

(Archives the revision with changeset 2 of myfile.pdf in 'c:\archived\_files'

folder. The archived file name will start with 'arch' (for example, arch\_11\_56).)

cm ^find "^revs ^where ^size > 26214400" --^format="{^item}#{^branch}" \

--^nototal | cm ^archive --^comment="volume00" --^file="volume00" -

(Archives all the files bigger than 25Mb on files starting with name

'volume00'.)

cm ^find "^revs ^where ^size > 26214400 ^and ^archived='true'" \

--^format="{^item}#{^branch}" --^nototal | cm ^archive --^restore

(Restores all the archived files bigger than 25Mb.)

== CMD\_DESCRIPTION\_ATTRIBUTE ==

Allows the user to manage attributes.

== CMD\_USAGE\_ATTRIBUTE ==

Usage:

cm ^attribute | ^att <command> [options]

Commands:

^create | ^mk

^delete | ^rm

^set

^unset

^rename

^edit

To get more information about each command run:

cm ^attribute <command> --^usage

cm ^attribute <command> --^help

== CMD\_HELP\_ATTRIBUTE ==

Examples:

cm ^attribute ^create status

cm ^attribute ^set ^att:status ^br:/main/SCM105 open

cm ^attribute ^unset ^att:status ^br:/main/SCM105

cm ^attribute ^delete ^att:status

cm ^attribute ^rename ^att:status "buildStatus"

cm ^attribute ^edit ^att:status "Status of the task in the CI pipeline"

== CMD\_DESCRIPTION\_CHANGELIST ==

Groups pending changes in changelists.

== CMD\_USAGE\_CHANGELIST ==

Usage:

a) Management of changelists objects:

cm ^changelist | ^clist [--^symlink]

(Displays all changelists.)

cm ^changelist | ^clist ^add <clist\_name>

[<clist\_desc>] [--^persistent | --^notpersistent] [--^symlink]

(Creates a changelist.)

cm ^changelist | ^clist ^rm <clist\_name> [--^symlink]

(Removes the selected changelist. If this changelist contains pending

changes, then these will be moved to the ^default changelist.)

cm ^changelist | ^clist ^edit <clist\_name> [<action\_name> <action\_value>]

[--^persistent | --^notpersistent] [--^symlink]

(Edits the selected changelist.)

b) Management of contents of a given changelist:

cm ^changelist | ^clist <clist\_name> (^add | ^rm) <path\_name>[ ...]

[--^symlink]

(Adds the selected changelist by adding ('^add') or removing ('^rm') the

change(s) that match with the given path\_name(s). Use a whitespace to

separate path\_names. Use double quotes (" ") to specify paths containing

spaces. The status of the paths must be '^Added' or '^Checked-out'.)

Options:

clist\_name The name of the changelist.

clist\_desc The description of the changelist.

action\_name Choose between '^rename' or '^description' to edit the

changelist.

action\_value Applies the new name or new description when editing

the changelist.

--^persistent The changelist will remain in the workspace even if its

contents are checked-in or reverted.

--^notpersistent (Default) The changelist will not remain in the

workspace even if its contents are checked-in or

reverted.

--^symlink Applies the operation to the symlink and not to the

target.

== CMD\_HELP\_CHANGELIST ==

Remarks:

The '^changelist' command handles both the workspace pending changelists and

the changes contained in a changelist.

Examples:

cm ^changelist

(Shows the current workspace changelists.)

cm ^changelist ^add config\_changes "dotConf files" --^persistent

(Creates a new changelist named 'config\_changes' and description 'dotConf

files' which will remain persistent in the current workspace once the

pending changelist is either checked-in or reverted.)

cm ^changelist ^edit config\_changes ^rename config\_files --^notpersistent

(Edits the changelist named 'config\_changes' and renames it to

'config\_files'. Also, it turns the changelist into "not persistent".)

cm ^changelist ^edit config\_changes --^notpersistent

(Edits the changelist named 'config\_changes' and turns it into "not persistent".)

cm ^changelist ^rm config\_files

(Removes the pending changelist 'config\_files' from the current workspace.)

cm ^changelist config\_files ^add foo.conf

(Adds the file 'foo.conf' to the 'config\_files' changelist.)

cm ^changelist config\_files ^rm foo.conf readme.txt

(Removes the files 'foo.conf' and 'readme.txt' from the 'config\_files'

changelist and moves the files to the system default changelist.)

== CMD\_DESCRIPTION\_CHANGESET ==

Executes advanced operations on changesets.

== CMD\_USAGE\_CHANGESET ==

Usage:

cm ^changeset <command> [options]

Commands:

^move | ^mv

^delete | ^rm

^editcomment | ^edit

To get more information about each command run:

cm ^changeset <command> --^usage

cm ^changeset <command> --^help

== CMD\_HELP\_CHANGESET ==

Examples:

cm ^changeset ^move ^cs:15@myrepo ^br:/main/scm005@myrepo

cm ^changeset ^delete ^cs:2b55f8aa-0b29-410f-b99c-60e573a309ca@devData

== CMD\_DESCRIPTION\_CHANGESET\_EDIT\_COMMENT ==

Modifies the comment of a changeset.

== CMD\_USAGE\_CHANGESET\_EDIT\_COMMENT ==

Usage:

cm ^changeset ^editcomment | ^edit <csetspec> <new\_comment>

Options:

csetspec The target changeset whose comment will be edited.

(Use 'cm ^help ^objectspec' to learn more about changeset

specs.)

new\_comment The new comment that will be added to the targeted

changeset.

== CMD\_HELP\_CHANGESET\_EDIT\_COMMENT ==

Remarks:

- The targeted changeset spec must be valid.

Examples:

cm ^changeset ^editcomment ^cs:15@myrepo "I forgot to add the checkin details"

cm ^changeset ^edit ^cs:cb11ecdb-1aa9-4f11-8698-dcab14e5885a \

"This comment text will replace the previous one."

== CMD\_DESCRIPTION\_CHANGESET\_MOVE ==

Moves a changeset and all its descendants to a different branch.

== CMD\_USAGE\_CHANGESET\_MOVE ==

Usage:

cm ^changeset ^move | ^mv <csetspec> <branchspec>

Options:

csetspec First changeset to be moved to a different branch. All

descendant changesets in the same branch will be

targeted by the command as well.

(Use 'cm ^help ^objectspec' to learn more about changeset

specs.)

branchspec The target branch where the targeted changesets are

stored. It needs to be empty or non-existing; if the

destination branch doesn't exist, it will be created by

the command.

(Use 'cm ^help ^objectspec' to learn more about branch

specs.)

== CMD\_HELP\_CHANGESET\_MOVE ==

Remarks:

- The targeted changeset spec must be valid.

- The destination branch must be either empty or non-existing.

- If the destination branch doesn't exist, it will created.

- Merge links will be kept unchanged, since branches don't affect them.

Examples:

cm ^changeset ^move ^cs:15@myrepo ^br:/main/scm005@myrepo

cm ^changeset ^move ^cs:cb11ecdb-1aa9-4f11-8698-dcab14e5885a ^br:/hotfix/TL-352

== CMD\_DESCRIPTION\_CHANGESET\_DELETE ==

Deletes a changeset from the repository.

== CMD\_USAGE\_CHANGESET\_DELETE ==

Usage:

cm ^changeset ^delete | ^rm <csetspec>

Options:

csetspec The target changeset to be removed. It must fulfill

some specific conditions. See Remarks for more info.

(Use 'cm ^help ^objectspec' to learn more about changeset

specs.)

== CMD\_HELP\_CHANGESET\_DELETE ==

Remarks:

- The target changeset must be the last in its branch.

- The target changeset cannot be the parent of any other changeset.

- The target changeset cannot be neither the source of a merge link nor

part of an interval merge as source.

- No label must be applied to the target changeset.

- The target changeset must not be the root changeset ('^cs:0')

Examples:

cm ^changeset ^rm ^cs:4525@myrepo@myserver

cm ^changeset ^delete ^cs:cb11ecdb-1aa9-4f11-8698-dcab14e5885a

== CMD\_DESCRIPTION\_CHANGEUSERPASSWORD ==

Changes the user password (UP).

== CMD\_USAGE\_CHANGEUSERPASSWORD ==

Usage:

cm ^changepassword | ^passwd

== CMD\_HELP\_CHANGEUSERPASSWORD ==

Remarks:

This command is only available when the security configuration is UP

(user/password). See the Administration guide for more information.

The old and new passwords are required.

Examples:

cm ^passwd

== CMD\_DESCRIPTION\_CHECKCONNECTION ==

Checks the connection to the server.

== CMD\_USAGE\_CHECKCONNECTION ==

Usage:

cm ^checkconnection | ^cc

== CMD\_HELP\_CHECKCONNECTION ==

Remarks:

- This command returns a message indicating whether there is a valid

connection to the configured Plastic SCM server.

- The command checks whether the configured user is valid or not. It also

checks the version compatibility with the server.

== CMD\_DESCRIPTION\_CHECKDB ==

Checks the repositories integrity.

== CMD\_USAGE\_CHECKDB ==

Usage:

cm ^checkdatabase | ^chkdb [<repserverspec> | <repspec>]

Use 'cm ^help ^objectspec' to learn more about repserver and rep specs.

== CMD\_HELP\_CHECKDB ==

Remarks:

- If neither repserverspec nor repspec are specified, the check will be

performed in the server specified in the client.conf file.

Examples:

cm ^checkdatabase ^repserver:localhost:8084

cm ^chkdb ^rep:default@localhost:8084

== CMD\_DESCRIPTION\_CHECKIN ==

Stores changes in the repository.

== CMD\_USAGE\_CHECKIN ==

Usage:

cm ^checkin | ^ci [<item\_path>[ ...]]

[-^c=<str\_comment> | -^commentsfile=<comments\_file>]

[--^all|-^a] [--^applychanged] [--^private] [--^update] [--^symlink]

[--^noshowchangeset]

[--^machinereadable [--^startlineseparator=<sep>]

[--^endlineseparator=<sep>] [--^fieldseparator=<sep>]]

Options:

item\_path Items to be checked-in. Use double quotes (" ") to

specify paths containing spaces. Use a whitespace to

separate item paths.

Use . to apply checkin to current directory.

-^c Applies the specified comment to the changeset created

in the checkin operation.

-^commentsfile Applies the comment in the specified file to the

changeset created in the checkin operation.

--^all | -^a The items changed, moved and deleted locally on the

given paths are also included.

--^applychanged Applies the checkin operation to the changed items

detected in the workspace along with the checked out

items.

--^private Private items detected in the workspace are also

included.

--^update Processes the update-merge automatically if it

eventually happens.

--^symlink Applies the checkin operation to the symlink and not

to the target.

--^noshowchangeset Doesn't print the result changeset.

--^machinereadable Outputs the result in an easy-to-parse format.

--^startlineseparator Used with the '--^machinereadable' flag, specifies how

the lines should start.

--^endlineseparator Used with the '--^machinereadable' flag, specifies how

the lines should end.

--^fieldseparator Used with the '--^machinereadable' flag, specifies how

the fields should be separated.

== CMD\_HELP\_CHECKIN ==

Remarks:

- If <item\_path> is not specified, the checkin involves all the

pending changes in the workspace.

- The checkin operation is always applied recursively from the given path.

- To checkin an item:

- The item must be under source code control.

- If the item is private (not under source code control), the --^private

flag is necessary in order to checkin it.

- The item must be checked out.

- If the item is changed but not checked out, the --^applychanged flag

is not necessary unless <item\_path> is a directory or it contains

wildcards ('\*').

Revision content should be different from previous revision in order to be

checked in.

Set PLASTICEDITOR environment variable to specify an editor to type the

comment.

Reading input from stdin:

The '^checkin' command can read paths from stdin. To do this, pass a single

dash "-".

Example: cm ^checkin -

Paths will be read until an empty line is entered.

This allows you to use pipe to specify which files to checkin.

Example:

dir /S /B \*.c | cm ^checkin --^all -

(In Windows, checkins all .c files in the workspace.)

Examples:

cm ^checkin file1.txt file2.txt

(Checkins the 'file1.txt' and 'file2.txt' checked-out files.)

cm ^checkin . -^commentsfile=mycomment.txt

(Checkins the current directory and sets the comment in the

'mycomment.txt' file.)

cm ^checkin link --^symlink

(Checkins the 'link' file and not the target; available on UNIX

environments.)

cm ^ci file1.txt -^c="my comment"

(Checkins 'file1.txt' and includes a comment.)

cm ^status --^short --^compact --^changelist=pending\_to\_review | cm ^checkin -

(Lists the paths in the changelist named 'pending\_to\_review' and redirects

this list to the input of the checkin command.)

cm ^ci . --^machinereadable

(Checkins the current directory, and prints the result in a simplified,

easier-to-parse format.)

cm ^ci . --^machinereadable --^startlineseparator=">" --^endlineseparator="<" --^fieldseparator=","

(Checkins the current directory, and prints the result in a simplified,

easier-to-parse format, starting and ending the lines, and

separating the fields with the specified strings.)

== CMD\_DESCRIPTION\_CHECKOUT ==

Marks files as ready to modify.

== CMD\_USAGE\_CHECKOUT ==

Usage:

cm ^checkout | ^co [<item\_path>[ ...]] [-^R | -^r | --^recursive]

[--^format=<str\_format>]

[--^errorformat=<str\_format>] [--^resultformat=<str\_format>]

[--^silent] [--^symlink] [--^ignorefailed]

[--^machinereadable [--^startlineseparator=<sep>]

[--^endlineseparator=<sep>] [--^fieldseparator=<sep>]]

Options:

item\_path Items to be checked-out. Use double quotes (" ") to

specify paths containing spaces. Use a whitespace to

separate item paths.

Use . to apply checkout to current directory.

-^R Checks out files recursively.

--^format Retrieves the output progress message in a specific

format. Check the examples for more information.

--^errorformat Retrieves the error message (if any) in a specific

format. Check the examples for more information.

--^resultformat Retrieves the output result message in a specific

format. Check the examples for more information.

--^silent Does not show any output at all.

--^symlink Applies the checkout operation to the symlink and not

to the target.

--^ignorefailed If an item cannot be locked (the exclusive checkout

cannot be performed), the checkout operation will

continue without it.

--^machinereadable Outputs the result in an easy-to-parse format.

--^startlineseparator Used with the '--^machinereadable' flag, specifies how

the lines should start.

--^endlineseparator Used with the '--^machinereadable' flag, specifies how

the lines should end.

--^fieldseparator Used with the '--^machinereadable' flag, specifies how

the fields should be separated.

== CMD\_HELP\_CHECKOUT ==

Remarks:

To checkout an item:

- The item must be under source code control.

- The item must be checked in.

If locks are configured on the server (lock.conf exists), then each time

a checkout on a path happens, Plastic checks if it meets any of the rules

and if so, the path will be in exclusive checkout (locked) so that none can

simultaneously checkout.

You can get all the locks in the server by using 'cm ^lock ^list'.

See the Administrator Guide for more information:

https://www.plasticscm.com/download/help/adminguide

The format string replaces the placeholder '{0}' with the path of the item

being checked out. Check the examples to see how to use it.

Reading input from stdin:

The '^checkout' command can read paths from stdin. To do this, pass a single

dash "-".

Example: cm ^checkout -

Paths will be read until an empty line is entered.

This allows you to use pipe to specify which files to checkout.

Example:

dir /S /B \*.c | cm ^checkout -

(In Windows, checkouts all .c files in the workspace.)

Examples:

cm ^checkout file1.txt file2.txt

(Checkouts 'file1.txt' and 'file2.txt' files.)

cm ^co \*.txt

(Checkouts all txt files.)

cm ^checkout .

(Checkouts current directory.)

cm ^checkout -^R c:\workspace\src

(Recursively checkouts 'src' folder.)

cm ^co file.txt --^format="Checking out item {0}"

--^errorformat="Error checking out item {0}" /

--^resultformat="Item {0} checked out"

(Checkouts 'file.txt' using the specified formatting strings

to show the progress, the result, and the errors of the operation.)

cm ^checkout link --^symlink

(Checkouts the 'link' file and not to the target; available on UNIX

environments.)

cm ^checkout . -^R --^ignorefailed

(Recursively checkouts the current folder, ignoring those files that can

not be checked out.)

cm ^co . --^machinereadable --^startlineseparator=">"

(Checkouts the current directory, and prints the result in a simplified,

easier-to-parse format, starting the lines with the specified strings.)

== CMD\_DESCRIPTION\_CHECKSELECTORSYNTAX ==

Checks the syntax of a selector.

== CMD\_USAGE\_CHECKSELECTORSYNTAX ==

Usage:

cm ^checkselectorsyntax | ^css --^file=<selector\_file>

(Checks the selector file syntax.)

^cat <selector\_file> | cm ^checkselectorsyntax | ^css -

(Unix. Checks selector file from standard input.)

^type <selector\_file> | cm ^checkselectorsyntax | ^css -

(Windows. Checks selector file from standard input.)

--^file The file to read a selector from.

== CMD\_HELP\_CHECKSELECTORSYNTAX ==

Remarks:

This command reads a selector on either a file or standard input, and

checks it for valid syntax. If the syntax check fails, the reason is

printed on standard output.

Examples:

cm ^checkselectorsyntax --^file=myselector.txt

(Checks the syntax of 'myselector.txt' file.)

^cat myselector.txt | cm ^checkselectorsyntax

(Checks the syntax of 'myselector.txt' from standard input.)

== CMD\_DESCRIPTION\_CHGREVTYPE ==

Changes an item revision type (binary or text).

== CMD\_USAGE\_CHGREVTYPE ==

Usage:

cm ^changerevisiontype | ^chgrevtype | ^crt <item\_path>[ ...] --^type=(^bin | ^txt)

item\_path Items to change revision type. Use double quotes (" ")

to specify paths containing spaces. Use a whitespace to

separate item paths.

--^type Target revisions type. Choose '^bin' or '^txt'.

== CMD\_HELP\_CHGREVTYPE ==

Remarks:

This command can only be applied to files, not directories.

The specified type must be a system supported one: '^bin' or '^txt' (binary

or text).

Examples:

cm ^changerevisiontype c:\workspace\file.txt --^type=^txt

(Changes 'file.txt' revision type to text.)

cm ^chgrevtype comp.zip "image file.jpg" --^type=^bin

(Changes 'comp.zip' and "image file.jpg" revision type to binary.)

cm ^crt \*.\* --^type=^txt

(Changes revision type of all files to text.)

== CMD\_DESCRIPTION\_TRIGGER\_EDIT ==

Edits a trigger.

== CMD\_USAGE\_TRIGGER\_EDIT ==

Usage:

cm ^trigger | ^tr ^edit <subtype\_type> <position\_number>

[--^position=<new\_position>]

[--^name=<new\_name>] [--^script=<script\_path>]

[--^filter=<str\_filter>] [--^server=<repserverspec>]

subtype\_type Trigger execution and trigger operation.

Type 'cm ^showtriggertypes' to see a list of trigger

types.

position\_number Position occupied by the trigger to be modified.

Options:

--^position New position of the specified trigger.

This position must not be in use by another

trigger of the same type.

--^name New name of the specified trigger.

--^script New execution path of the specified trigger script.

If the script starts with "^webtrigger ", it will be

considered as a web trigger. See Remarks for more

further details.

--^filter Checks only items that match the specified filter.

--^server Modifies the trigger on the specified server.

If no server is specified, executes the command on the

one configured on the client.

(Use 'cm ^help ^objectspec' to learn more about server

specs.)

== CMD\_HELP\_TRIGGER\_EDIT ==

Remarks:

Web triggers: A web trigger is created by typing "^webtrigger <target-uri>"

as the trigger command. In this case, the trigger will execute a POST query

against the specified URI, where the request body contains a JSON

dictionary with the trigger environment variables, and a fixed INPUT key

pointing to an array of strings.

Examples:

cm ^trigger ^edit ^after-setselector 6 --^name="Backup2 manager" --^script="/new/path/al/script"

cm ^tr ^edit ^before-mklabel 7 --^position=4 --^server=myserver:8084

cm ^trigger ^edit ^after-add 2 --^script="^webtrigger http://myserver.org/api"

== CMD\_DESCRIPTION\_CODEREVIEW ==

Creates, edits, or deletes code reviews.

== CMD\_USAGE\_CODEREVIEW ==

Usage:

cm ^codereview <spec> <title> [--^status=<status\_name>]

[--^assignee=<user\_name>] [--^format=<str\_format>]

[--^repository=<rep\_spec>]

(Creates a code review.)

cm ^codereview -^e <id> [--^status=<status\_name>] [--^assignee=<user\_name>]

[--^repository=<rep\_spec>]

(Edits a code review.)

cm ^codereview -^d <id> [ ...] [--^repository=<rep\_spec>]

(Deletes one or more code reviews.)

spec It can be either a changeset spec or a branch spec. It

will be the target of the new code review. (Use

'cm ^help ^objectspec' to learn more about changeset or

branch specs.)

title A text string to be used as title of the new

code review.

id The code review identification number. A GUID can be

used as well.

Options:

-^e Edits the parameters of an existing code review.

-^d Deletes one or more existing code reviews. Use a

whitespace to separate the code reviews IDs.

--^status Sets the new status of a code review. See Remarks

for additional information.

--^assignee Sets the new assignee of a code review.

--^format Retrieves the output message in a specific format. See

Remarks for additional information.

--^repository Sets the repository to be used as default. (Use

'cm ^help ^objectspec' to learn more about repository

specs.)

== CMD\_HELP\_CODEREVIEW ==

Remarks:

This command allows users to manage code reviews: create, edit, and delete

code reviews for changesets or branches.

To create a new code review, a changeset/branch spec and a title are

required. The initial status and assignee can be set, too. An ID (or GUID

if requested) will be returned as a result.

To edit or delete an existing code review, the target code review ID

(or GUID) is required. No messages are displayed if there are no errors.

The status parameter can only be one of the following: ^"Under review"

(default), ^"Reviewed", or ^"Rework required".

The repository parameter is available to set the default working

repository. This is useful when the user wants to manage reviews on

a server different than the one associated to the current workspace, or

when there is no current workspace at all.

Output format customization:

This command accepts a format string to show the output.

The output parameters of this command are the following:

{0} id

{1} guid

Please note that the '--^format' parameter only takes effect when creating

a new code review.

Examples:

cm ^codereview ^cs:1856@myrepo@myserver:8084 "My code review" --^assignee=dummy

cm ^codereview ^br:/main/task001@myrepo@myserver:8084 "My code review" \

--^status=^"Rework required" --^assignee=newbie --^format="{^id} -> {^guid}"

cm ^codereview 1367 -^e --^assignee=new\_assignee

cm ^codereview -^e 27658884-5dcc-49b7-b0ef-a5760ae740a3 --^status=Reviewed

cm ^codereview -^d 1367 --^repository=myremoterepo@myremoteserver:18084

cm ^codereview 27658884-5dcc-49b7-b0ef-a5760ae740a3 -^d

== CMD\_DESCRIPTION\_CRYPT ==

Encrypts a password.

== CMD\_USAGE\_CRYPT ==

Usage:

cm ^crypt <mypassword>

mypassword Password to be encrypted.

== CMD\_HELP\_CRYPT ==

Remarks:

This command encrypts a given password passed as argument.

It is designed to encrypt passwords in configuration files and increase

safety.

Examples:

cm ^crypt dbconfpassword -> ENCRYPTED: encrypteddbconfpassword

(Encrypts the password in the database configuration file: 'db.conf'.)

== CMD\_DESCRIPTION\_DEACTIVATEUSER ==

Deactivates a licensed user.

== CMD\_USAGE\_DEACTIVATEUSER ==

Usage:

cm ^deactivateuser | ^du <usr\_name>[ ...] [--^server=<name:port>]

[--^nosolveuser]

usr\_name The user name(s) to deactivate. Use a whitespace to

separate user names.

If SID, then '--^nosolveuser' is required.

Options:

--^server Deactivates the user on the specified server.

If no server is specified, executes the command on the

one configured on the client.

--^nosolveuser With this option, the command will not check whether

the user name exists on the authentication system. The

<usr\_name> must be a user SID.

== CMD\_HELP\_DEACTIVATEUSER ==

Remarks:

This command sets a user to inactive, disabling the usage of Plastic SCM

for that user.

See the 'cm ^activateuser' command for more information about activating

Plastic SCM users.

This command checks whether the user exists on the underlying authentication

system (e.g. ActiveDirectory, LDAP, User/Password...).

To force the deactivation of a user that no longer exists on the

authentication system, you can use the '--^nosolveuser' option.

Examples:

cm ^deactivateuser john

cm ^du peter "mary collins"

cm ^deactivateuser john --^server=myserver:8084

cm ^deactivateuser S-1-5-21-3631250224-3045023395-1892523819-1107 --^nosolveuser

== CMD\_DESCRIPTION\_DIFF ==

Shows differences between files, changesets, and labels.

== CMD\_USAGE\_DIFF ==

Usage:

cm ^diff <csetspec> | <lbspec> | <shspec> [<csetspec> | <lbspec> | <shspec>]

[<path>]

[--^added] [--^changed] [--^moved] [--^deleted]

[--^repositorypaths] [--^download=<download\_path>]

[--^encoding=<name>]

[--^ignore=(^eol | ^whitespaces | ^"eol&whitespaces" | ^none)]

[--^clean]

[--^format=<str\_format>] [--^dateformat=<str\_format>]

Shows differences between a 'source' changeset or shelveset, and a

'destination' changeset or shelveset. The changesets can be specified

using either a changeset or label specification.

Where two specifications are given, the first will be the 'source' of

the diff; the second, the 'destination'.

If only one specification is given, the 'source' will be the parent

changeset of the specified 'destination'.

If an optional path is specified, the Diff Window will launch to show

differences between the two revision of that file.

cm ^diff <revspec1> <revspec2>

Shows differences between a pair of revisions. The differences are

shown in the Diff Window. The first revision specified will appear on

the left.

cm ^diff <brspec> [--^added] [--^changed] [--^moved] [--^deleted]

[--^repositorypaths] [--^download=<download\_path>]

[--^encoding=<name>]

[--^ignore=(^eol | ^whitespaces | ^"eol&whitespaces" | ^none)]

[--^clean]

[--^format=<str\_format>] [--^dateformat=<str\_format>]

[--^fullpaths | --^fp]

Shows the branch differences.

(Use 'cm ^help ^objectspec' to learn more about specs.)

Options:

--^added Prints only differences consisting of items added to

the repository.

--^changed Prints only differences consisting of items that

changed.

--^moved Prints only differences consisting of moved or renamed

items.

--^deleted Prints only differences consisting of items that were

deleted.

If '--^added', '--^changed', '--^moved' or '--^deleted' are

not specified, then the command prints all differences.

'^A' means added items.

'^C' means changed items.

'^D' means deleted items.

'^M' means moved items. The left item is the original,

the right is the destination.

--^repositorypaths Prints repository paths instead of workspace paths.

(This option overrides the '--^fullpaths' option.)

--^download Stores the differences content in the specified output

path.

--^encoding Specifies the output encoding, i.e.: utf-8.

See the MSDN documentation at

http://msdn.microsoft.com/en-us/library/system.text.encoding.aspx

to get the table of supported encodings and its format,

(at the end of the page, in the "Name" column).

--^ignore Sets the specified comparison method.

See Remarks for more info.

--^clean Does not take into account the differences generated

because of a merge, but only the differences created by

simple checkins.

--^format Retrieves the output message in a specific format. See

Remarks for more info.

--^dateformat Format used to output dates.

--^fullpaths, --^fp Forces printing full workspace paths for files and

directories when possible.

== CMD\_HELP\_DIFF ==

Remarks:

Comparison methods:

^eol Ignores the end of line differences.

^whitespaces Ignores the whitespace differences.

^"eol&whitespaces" Ignores the end of line and whitespace differences.

^none Detects the end of line and whitespace differences.

This command accepts a format string to show the output.

The parameters of this command are the following:

{^path} Item path.

{^date} Change date/time.

{^owner} Change author.

{^revid} Revision id of the revision considered as the

destination in the diff.

{^parentrevid} Revision id of the parent of the revision considered

as the destination of the diff.

{^baserevid} Revision id of the revision considered as the source

in the diff.

{^srccmpath} Server path before moving the item (move operation).

{^dstcmpath} Server path after moving the item (move operation).

{^type} Item type:

^D directory,

^B binary file,

^F text file,

^S symlink,

^X Xlink.

{^repository} Repository of the item.

{^status} Item status:

^A added,

^D deleted,

^M moved,

^C changed.

{^fsprotection} Shows item permissions (Linux/Mac chmod).

{^srcfsprotection} Shows parent revision item permissions.

{^newline} Inserts a new line.

Notes on '^revid':

For added items, the '^baserevid' and '^parentrevid' will be -1, as no

previous revision exists in this case.

For deleted items, the '^revid' is the id of the source revision, and the

'^baserevid' will be -1, as there is no destination revision.

For Xlinks, both '^baserevid' and '^parentrevid' are always -1.

Examples:

Comparing branches:

cm ^diff ^br:/main/task001

cm ^diff ^br:/main/task001 \doc\readme.txt

Comparing changeset trees:

cm ^diff 19

cm ^diff 19 25

cm ^diff ^cs:19 ^cs:25 --^format="{^path} {^parentrevid}"

cm ^diff ^cs:19 ^cs:23 --^format="{^date} {^path}" --^dateformat="yy/dd/MM HH:mm:ss"

cm ^diff ^cs:19 ^cs:23 --^changed

cm ^diff ^cs:19 ^cs:23 --^repositorypaths

cm ^diff ^cs:19 ^cs:23 --^download="D:\temp"

cm ^diff ^cs:19 ^cs:23 --^clean

cm ^diff ^cs:19 ^cs:23 \doc\readme.txt

Comparing label trees:

cm ^diff ^lb:FirstReleaseLabel ^lb:SecondReleaseLabel

cm ^diff ^lb:tag\_193.2 ^cs:34214

cm ^diff ^cs:31492 ^lb:tag\_193.2

Comparing shelve trees:

cm ^diff ^sh:2

cm ^diff ^sh:2 ^sh:4

Comparing revspecs:

cm ^diff ^rev:readme.txt#^cs:19 ^rev:readme.txt#^cs:20

cm ^diff ^serverpath:/doc/readme.txt#^cs:19@myrepo \

^serverpath:/doc/readme.txt#^br:/main@myrepo@localhost:8084

== CMD\_DESCRIPTION\_DIFFMETRICS ==

Shows diff metrics between two revs.

== CMD\_USAGE\_DIFFMETRICS ==

Usage:

cm ^diffmetrics | ^dm <revspec1> <revspec2> [--^format=<str\_format>]

[--^encoding=<name>]

[--^ignore=(^eol | ^whitespaces | ^"eol&whitespaces" | ^none)]

revspec Revisions used to compare.

(Use 'cm ^help ^objectspec' to learn more about rev specs.)

Options:

--^format Retrieves the output message in a specific format. See

Remarks for more info.

--^encoding Specifies the output encoding, i.e.: utf-8.

See the MSDN documentation at

http://msdn.microsoft.com/en-us/library/system.text.encoding.aspx

to get the table of supported encodings and its format,

(at the end of the page, in the "Name" column).

--^ignore Sets the specified comparison method.

See Remarks for more info.

== CMD\_HELP\_DIFFMETRICS ==

Remarks:

The metrics are: number of changed, added, and deleted lines.

This command accepts a format string to show the output.

The output parameters of this command are the following:

{0} Number of changed lines.

{1} Number of added lines.

{2} Number of deleted lines.

Examples:

cm ^diffmetrics file.txt#^cs:2 file.txt#^br:/main/scm0211 \

--^format="There are {0} changed, {1} added and {2} deleted lines."

(Retrieves diffmetrics results formatted.)

cm ^dm file.txt#^cs:2 file.txt#^cs:3 --^encoding=utf-8 --^ignore=^whitespaces

== CMD\_DESCRIPTION\_FASTEXPORT ==

Exports a repository in fast-export format.

== CMD\_USAGE\_FASTEXPORT ==

Usage:

cm ^fast-export | ^fe <repspec> <fast-export-file>

[--^import-marks=<marks\_file>]

[--^export-marks=<marks\_file>]

[--^branchseparator=<chr\_separator>]

[--^nodata] [--^from=<changesetid>] [--^to=<changesetid>]

Options:

repspec The repository which the data will be exported from.

(Use 'cm ^help ^objectspec' to learn more about rep specs.)

fast-export-file The file with the repository data in Git fast-export

format.

--^import-marks The marks file used for incremental imports. This file

has been previously exported by '--^export-marks'. The

changesets described in this file will not be imported

because they were already in a previous import.

--^export-marks The file where the imported changesets will be saved.

This file is used in a later fast-import to signal the

changesets that have been already imported.

--^branchseparator Plastic SCM uses "/" as default separator in the branch

hierarchy. This option allows using char as a hierarchy

separator, so main-task-sub would be mapped in Plastic

SCM as /main/task/sub.

--^nodata Exports the repository without including the data. This

is useful to check if the export will run correctly.

--^from Exports from a particular changeset.

--^to Exports to a particular changeset.

== CMD\_HELP\_FASTEXPORT ==

Remarks:

- In order to import a Plastic SCM repository to Git, use a command such as:

^cat repo.fe.00 | ^git ^fast-import --^export-marks=marks.git --^import-marks=marks.git

- Incremental export is supported using a marks file that contains the

changesets previously imported ('--^import-marks' and '--^export-marks'

files).

This means that only the new changesets that were not exported in the

previous fast-export will be exported.

Examples:

cm ^fast-export repo@localhost:8087 repo.fe.00 --^import-marks=marks.cm \

--^export-marks=marks.cm

(Exports the repository 'repo' in the local server into the 'repo.fe.00'

file in Git fast-export format and creates the marks files to perform

incremental exports later.)

cm ^fast-export repo@localhost:8087 repo.fe.00 --^from=20

(Exports the repository 'repo' in the local server into the 'repo.fe.00'

file in Git fast-export format from changeset '20'.)

== CMD\_DESCRIPTION\_FASTIMPORT ==

Imports Git fast-export data into a repository.

== CMD\_USAGE\_FASTIMPORT ==

Usage:

cm ^fast-import | ^fi <repspec> <fast-export-file>

[--^import-marks=<marks\_file>]

[--^export-marks=<marks\_file>]

[--^stats] [--^branchseparator=<chr\_separator>]

[--^nodata] [--^ignoremissingchangesets] [--^mastertomain]

Options:

repspec The repository into which the data will be

imported. It is created if it did not previously

exist. (Use 'cm ^help ^objectspec' to learn more

about rep specs.)

fast-export-file The file with the repository data in Git

fast-export format.

--^import-marks The marks file used for incremental imports.

This file has been previously exported by

'--^export-marks'. The changesets described in

this file wont be imported because they

were already in a previous import.

--^export-marks The file where the imported changesets will

be saved. This file is used in a later

fast-import to signal the changesets that have

been already imported.

--^stats Prints some statistics about the import process.

--^branchseparator Plastic SCM uses "/" as default separator in

the branch hierarchy. This option allows using

char as a hierarchy separator, so main-task-sub

would be mapped in Plastic SCM as /main/task/sub.

--^nodata Imports Git fast-export without including the

data. This is useful to check if the import will

run correctly.

--^ignoremissingchangesets Any changesets that cannot be imported are

discarded and the fast-import operation

continues without them.

--^mastertomain Imports using "^main" instead of "^master".

== CMD\_HELP\_FASTIMPORT ==

Remarks:

- In order to export a git repository, use a command such as:

^git ^fast-export --^all -^M --^signed-tags=^strip --^tag-of-filtered-object=^drop> ..\git-fast-export.dat

The -^M option is important to detect moved items.

- The specified repository is created in case it did not exist.

- Incremental import is supported using a marks file that contains the

changesets previously imported ('--^import-marks' and '--^export-marks'

files).

This means that only the new changesets that were not imported in the

previous fast-import will be imported.

Examples:

cm ^fast-import mynewrepo@atenea:8084 repo.fast-export

(Imports the contents exported in the 'repo.fast-export' file into

'mynewrepo' repository on server 'atenea:8084'.)

cm ^fast-import repo@atenea:8084 repo.fast-export --^export-marks=rep.marks

(Imports the contents exported in the 'repo.fast-export' file into

'repo' repository on server 'atenea:8084' and creates a marks file

to perform incremental imports later.)

cm ^fast-import repo@server:8084 repo.fast-export --^import-marks=repo.marks \

--^export-marks=repo.marks

(Imports the contents of the 'repo.fast-export' file. Only the new

changesets that were not in the marks file are imported. The same marks

file is used to save the list of changesets again for the next

incremental import.)

== CMD\_DESCRIPTION\_FILEINFO ==

Retrieves detailed information about the items in the workspace.

== CMD\_USAGE\_FILEINFO ==

Usage:

cm ^fileinfo <item\_path>[ ...] [--^fields=<field\_value>[,...]]

[[--^xml | -^x [=<output\_file>]] | [--^format=<str\_format>]]

[--^symlink] [--^encoding=<name>]

item\_path Items to display. Use a whitespace to separate the

items.

Use double quotes (" ") to specify paths containing

spaces.

Options:

--^fields A string of comma-separated values. This selects which

fields will be printed for each item. See Remarks for

more information.

--^xml | -^x Prints the output in XML format to the standard output.

It is possible to specify an output file. This option

cannot be combined with '--^format'.

--^format Retrieves the output message in a specific format. See

Remarks for more info. This option cannot be combined

with '--^xml'.

This '--^format' option prevails over '--^fields' if both

are specified.

--^symlink Applies the fileinfo operation to the symlink and not

to the target.

--^encoding Specifies the output encoding, i.e.: utf-8.

See the MSDN documentation at

http://msdn.microsoft.com/en-us/library/system.text.encoding.aspx

to get the table of supported encodings and its format,

(at the end of the page, in the "Name" column).

== CMD\_HELP\_FILEINFO ==

Remarks:

This command prints a detailed list of attributes for each selected item.

Each attribute is printed on a new line by default.

The attribute list can be modified to display only the attributes the user

needs. This can be achieved using the '--^fields=<field\_list>', which accepts

a string of comma-separated attribute names. This way, only those arguments

whose name has been indicated are shown.

Revision head changeset:

This option is disabled by default. Please note that retrieving this

attribute is significantly slower than the rest of them, so we advise users

to group together as many items as possible. This will improve execution

times by avoiding many separate 'cm ^fileinfo' executions.

Also, this feature is not currently available for controlled directories.

You can find below the complete list of available attribute names.

Names marked with an asterisk ('\*') will not be shown by default:

^ClientPath The local path on disk for the item.

^RelativePath The workspace-relative path.

^ServerPath The repository path for the item.

(Note: Transformed workspaces are not

currently supported for this option).

^Size Item size.

^Hash Item hash sum.

^Owner The user the item belongs to.

^RevisionHeadChangeset (\*) The changeset of the revision loaded in the

head changeset of the branch.

(Please see note above.)

^RevisionChangeset The changeset of the revision currently loaded

in the workspace.

^RepSpec The repository specification for the item.

(Use 'cm ^help ^objectspec' to learn more about

rep specs.)

^Status The workspace item status: added, checked out,

deleted, etc.

^Type Revision type (text, binary, directory, symlink,

or unknown).

^Changelist The changelist the item belongs to (if any).

^IsLocked (\*) Whether the item is locked by exclusive

checkout or not.

^LockedBy (\*) The user who exclusively checked out the item.

^LockedWhere (\*) The location where the item was exclusively

checked out.

^IsUnderXlink Whether the item is located under an Xlink

or not.

^UnderXlinkTarget The target of the Xlink the item is under

(if any).

^UnderXlinkPath The item server path in the Xlinked repository

(if any).

^UnderXlinkWritable Whether the Xlink the item belongs to is

writable or not.

^UnderXlinkRelative Whether the Xlink the items belongs to is

relative or not.

^IsXlink Whether the item itself is a Xlink or not.

^XlinkTarget The target repository the item points to, if it

is a Xlink.

^XlinkName The Xlink name of the item, if it is

actually one.

^XlinkWritable Whether the Xlink item is a writable Xlink

or not.

^XlinkRelative Whether the Xlink item is a relative Xlink

or not.

Output format customization:

This command accepts a format string to show the output.

The output parameters of this command are the following:

{^ClientPath}

{^RelativePath}

{^ServerPath}

{^Size}

{^Hash}

{^Owner}

{^RevisionHeadChangeset}

{^RevisionChangeset}

{^Status}

{^Type}

{^Changelist}

{^IsLocked}

{^LockedBy}

{^LockedWhere}

{^IsUnderXlink}

{^UnderXlinkTarget}

{^UnderXlinkPath}

{^UnderXlinkWritable}

{^UnderXlinkRelative}

{^IsXlink}

{^XlinkTarget}

{^XlinkName}

{^XlinkWritable}

{^XlinkRelative}

{^RepSpec}

Please note that '--^format' and '--^xml' options are mutually exclusive, so

they can't be used at the same time.

Examples:

cm ^fileinfo file1.txt file2.txt dir/

cm ^fileinfo "New Project.csproj" --^xml

cm ^fileinfo assets.art --^fields=^ServerPath,^Size,^IsLocked,^LockedBy

cm ^fileinfo proj\_specs.docx --^fields=^ServerPath,^RevisionChangeset --^xml

cm ^fileinfo samples.ogg --^format="{^ServerPath}[{^Owner}] -> {^Size}"

== CMD\_DESCRIPTION\_FIND\_QUERY ==

Runs SQL-like queries to find Plastic SCM objects.

== CMD\_USAGE\_FIND\_QUERY ==

Usage:

cm ^find <object\_type>

[^where <str\_conditions>]

[^on ^repository '<repspec>' | ^on ^repositories '<repspec1>','<repspec2>'[,...]]

[--^format=<str\_format>] [--^dateformat=<date\_format>]

[--^nototal] [--^file=<dump\_file>] [--^xml]

[--^encoding=<name>]

object\_type Object type to find.

Use 'cm ^help ^showfindobjects' to learn how to specify

these objects.

You can also read the 'cm ^find' guide:

https://www.plasticscm.com/download/help/cmfind

Options:

str\_conditions Searches conditions on an object attributes.

repspec Searches repositories alias or specification.

In the case of '^on ^repositories', use a comma to

separate the repspec fields.

(Use 'cm ^help ^objectspec' to learn more about repository

specifications.)

--^format Retrieves the output message in a specific format.

Read the 'cm ^find' guide to see all the object

attributes that can be used as output format strings:

https://www.plasticscm.com/download/help/cmfind

--^dateformat Format used to output dates.

--^nototal Does not output record count at the end.

--^file File to dump results.

--^xml Prints the output in XML format to the standard output.

--^encoding Specifies the output encoding, i.e.: utf-8.

See the MSDN documentation at

http://msdn.microsoft.com/en-us/library/system.text.encoding.aspx

to get the table of supported encodings and its format,

(at the end of the page, in the "Name" column).

== CMD\_HELP\_FIND\_QUERY ==

Remarks:

If no repository is specified, the search is made on the repository

configured in the workspace.

When you run queries using comparison operators (>, <, >=, <=) from the

command line, remember that the shell considers these operators as IO

redirections. So you will need to enclose the queries in double quotation

marks.

The 'cm ^find' command accepts a format string to show the output.

Each output parameter is identified by a string and the user can refer it

by typing the parameter number between '{' and '}' brackets.

Output parameters usually correspond to the attributes of the object.

These are some valid output format strings:

--^format={^id}{^date}{^name}

--^format="{^item}#{^branch} ^with ^date {^date}"

XML and encoding considerations:

When the '--^xml' option is specified, the command shows the command result

as an XML text in the standard output. The operating system default encoding

is used to show the text, so it is possible that not-ANSI characters are

incorrectly visualized in the console. If you redirect the command output to

a file, it will be correctly visualized. When both '--^xml' and '--^file'

options are specified, the default encoding will be utf-8.

Examples:

cm ^find ^revision

cm ^find ^revision "^where ^changeset=23 ^and ^owner='maria'"

cm ^find ^branch "^on ^repository 'rep1'"

cm ^find ^label "^on ^repositories 'rep1', '^rep:default@localhost:8084'"

cm ^find ^branch "^where ^parent='^br:/main' ^on ^repository 'rep1'"

cm ^find ^revision "^where ^item='^item:.'" --^format="{^item}#{^branch}"

cm ^find ^revision "^where ^item='^item:.'" --^xml --^file=c:\queryresults\revs.xml

== CMD\_DESCRIPTION\_FINDCHANGED ==

Gets a list of changed files. This command is deprecated and kept just for

backwards compatibility. Use 'cm ^status' instead.

== CMD\_USAGE\_FINDCHANGED ==

Usage:

cm ^findchanged | ^fc [-^R | -^r | --^recursive] [--^checkcontent]

[--^onlychanged] [<path>]

Options:

-^R Recursively finds in directories.

--^checkcontent Compares files by content.

--^onlychanged Finds only changed files; checkouts will not be

obtained.

path (Default: current directory.)

Initial path to find changed files.

== CMD\_HELP\_FINDCHANGED ==

Remarks:

If no '--^checkcontent' option is given, Plastic finds changes based on the

file timestamp. When '--^checkcontent' option is specified, the file or

folder contents are compared, instead of using the timestamp.

This command is useful to detect changed files while disconnected from

the Plastic SCM server. The output can be piped to the checkout command,

to check the changes later (see examples).

Examples:

cm ^findchanged .

(Finds changed files in the current directory.)

cm ^findchanged -^R . | cm ^checkout -

(Checkouts changed elements.)

== CMD\_DESCRIPTION\_FINDCHECKEDOUT ==

Gets a list of checked out items. This command is deprecated and kept just for

backwards compatibility. Use 'cm ^status' instead.

== CMD\_USAGE\_FINDCHECKEDOUT ==

Usage:

cm ^findcheckouts | ^fco [--^format=<str\_format>] [--^basepath]

Options:

--^format Retrieves the output message in a specific format. See

Remarks for more info.

--^basepath The path to start searching checkouts from. If not

specified, the current path is used.

== CMD\_HELP\_FINDCHECKEDOUT ==

Remarks:

This command is useful to checkin or undocheckout all checked out items in

one single step, redirecting the standard output to other command.

See examples.

This command accepts a format string to show the output.

The output parameters of this command are the following:

{0} Date.

{1} Owner.

{2} Workspace info.

{3} Client machine name.

{4} Item path.

{5} Branch and repository info.

Examples:

cm ^findcheckouts --^format="File {4} changed on branch {5}"

(Finds checked out items and formats the output with file path and branch

and repository info.)

cm ^findcheckouts --^format={4} | cm ^checkin -

(Checkins all checked out items.)

cm ^findcheckouts --^format={4} | cm ^undocheckout -

(Undocheckouts of all checked out items.)

== CMD\_DESCRIPTION\_FINDPRIVATE ==

Gets a list of private items. This command is deprecated and kept just for

backwards compatibility. Use 'cm ^status' instead.

== CMD\_USAGE\_FINDPRIVATE ==

Usage:

cm ^findprivate | ^fp [-^R | -^r | --^recursive] [--^exclusions] [<path>]

Options:

-^R Recursively finds in directories.

--^exclusions This option allows cutting the search inside the ignored

paths, defined by the file ignore.conf.

path (Default: current directory.)

Initial path to find private files.

== CMD\_HELP\_FINDPRIVATE ==

Remarks:

If any path is specified, Plastic SCM will begin searching from the

current directory.

This command is useful to add private items on a folder, piping the output

to the add command. See examples.

Examples:

cm ^findprivate .

cm ^findprivate -^R | cm ^add -

(Recursively searches private items and add them.)

== CMD\_DESCRIPTION\_GETCONFIG ==

Obtains configuration info.

== CMD\_USAGE\_GETCONFIG ==

Usage:

cm ^getconfig [^setfileasreadonly] [^location] [^extensionworkingmode]

[^extensionprefix] [^defaultrepserver]

^setfileasreadonly Returns whether the protected files are left as

read-only or not.

^location Returns the client config path.

^extensionworkingmode Returns the extension working mode.

^extensionprefix Returns the configured extension prefix.

^defaultrepserver Returns the location of the default repository

server.

== CMD\_HELP\_GETCONFIG ==

Examples:

cm ^getconfig ^setfileasreadonly

== CMD\_DESCRIPTION\_GETFILE ==

Downloads the content of a given revision.

== CMD\_USAGE\_GETFILE ==

Usage:

cm ^getfile | ^cat <revspec> [--^file=<output\_file>] [--^debug]

[--^symlink] [--^raw]

revspec Object specification. (Use 'cm ^help ^objectspec' to learn

more about specs.)

Options:

--^file File to save the output. By default, it is printed on the

standard output.

--^debug When a directory specification is used, the command

shows all the items in the directory, its revision id

and file system protection.

--^symlink Applies the operation to the symlink and not to the

target.

--^raw Displays the raw data of the file.

== CMD\_HELP\_GETFILE ==

Examples:

cm ^cat myfile.txt#^br:/main

(Obtains the last revision in branch '^br:/main' of 'myfile.txt'.)

cm ^getfile myfile.txt#^cs:3 --^file=tmp.txt

(Obtains the changeset 3 of 'myfile.txt' and write it to file 'tmp.txt'.)

cm ^cat ^serverpath:/src/foo.c#^br:/main/task003@myrepo

(Obtains the contents of '/src/foo.c' at the last changeset of branch

'/main/task003' in repository 'myrepo')

cm ^cat ^revid:1230@^rep:myrep@^repserver:myserver:8084

(Obtains the revision with id 1230.)

cm ^getfile ^rev:info\ --^debug

(Obtains all revisions in the 'info' directory.)

== CMD\_DESCRIPTION\_GETREVISION ==

Loads a revision in the workspace.

== CMD\_USAGE\_GETREVISION ==

This command modifies the revision loaded in the workspace, so it can affect

future merges.

It is an advanced command inherited from old versions, so use it with care.

Usage:

cm ^getrevision <revspec>

revspec Object specification. (Use 'cm ^help ^objectspec' to learn

more about rev specs.)

== CMD\_HELP\_GETREVISION ==

Examples:

cm ^getrevision file.txt#^cs:3

(Gets changeset 3 revision of 'file.txt'.)

== CMD\_DESCRIPTION\_GETSTATUS ==

Gets the status of an item.

== CMD\_USAGE\_GETSTATUS ==

This is an automation command, meant to be used to automate 'cm' only.

It is not as user friendly as it should be.

Usage:

cm ^getstatus | ^gs <item\_path>[ ...] [--^format=<str\_format>] [--^stats]

[-^R | -^r | --^recursive]

item\_path Item or items to get status from. Use double quotes

(" ") to specify paths containing spaces. Use a

whitespace to separate paths.

Options:

--^format Retrieves the output message in a specific format. See

Remarks for more info.

--^stats Prints some statistics about the get status process.

-^R Shows recursively the status in directories.

== CMD\_HELP\_GETSTATUS ==

Remarks:

This command accepts a format string to show the output.

The output parameters of this command are the following:

{0} Item path.

{1} Item status:

0 private,

1 checked in,

2 checked out.

Reading input from stdin:

The '^getstatus' command can read paths from stdin. To do this, pass a

single dash "-".

Example: cm ^getstatus -

Paths will be read until an empty line is entered.

This allows you to use pipe to specify which paths to get the status for.

Example:

dir /S /B \*.c | cm ^getstatus --^format="Path {0} Status {1}" -

(In Windows, gets the status of all .c files in the workspace.)

Examples:

cm ^getstatus file1.txt file2.txt

(Gets the status of the files.)

cm ^gs info\ -^R --^format="The item {0} has the status {1}"

(Gets the status of the directory and all of its items and shows a

formatted output.)

== CMD\_DESCRIPTION\_GETTASKBRANCHES ==

Gets branches linked with a task.

== CMD\_USAGE\_GETTASKBRANCHES ==

This is an automation command, meant to be used to automate 'cm' only.

It is not as user friendly as it should be.

Usage:

cm ^gettaskbranches | ^gtb <task\_name> [--^format=<str\_format>]

[--^dateformat=<date\_format>]

task\_name The task identifier.

Options:

--^format Retrieves the output message in a specific format. See

Remarks for more info.

--^dateformat Format used to output dates.

== CMD\_HELP\_GETTASKBRANCHES ==

Remarks:

This command accepts a format string to show the output.

The output parameters of this command are the following:

{^tab} Inserts a tab space.

{^newline} Inserts a new line.

{^name} Branch name.

{^owner} Owner of the branch.

{^date} Date when the branch was created.

{^type} Branch type ('T' if it is smart or 'F' if not).

{^parent} Parent branch.

{^comment} Comment of the branch.

{^repname} Repository where the branch exists.

{^repserver} Server name.

Examples:

cm ^gettaskbranches 4311

cm ^gtb 4311 --^format="^br:{^name}"

cm ^gtb 4311 --^format="^br:{^name} {^date}" --^dateformat="yyyy/MM/dd HH:mm:ss"

== CMD\_DESCRIPTION\_GETWWI ==

Shows info about the workspace selector.

== CMD\_USAGE\_GETWWI ==

Usage:

cm ^wi [<wk\_path>]

Options:

wk\_path Path of a workspace on the machine.

== CMD\_HELP\_GETWWI ==

Remarks:

The '^wi' command shows the working configuration of a workspace (repository,

branch, and/or label).

Examples:

cm ^wi c:\mywk

== CMD\_DESCRIPTION\_GWP ==

Gets workspace info from a path.

== CMD\_USAGE\_GWP ==

This is an automation command, meant to be used to automate 'cm' only.

It is not as user friendly as it should be.

Usage:

cm ^getworkspacefrompath | ^gwp <item\_path> [--^format=<str\_format>]

item\_path File or folder on disk.

Options:

--^format Retrieves the output message in a specific format. See

Remarks for more info.

== CMD\_HELP\_GWP ==

Remarks:

This command shows information about the workspace that is located in path.

This command accepts a format string to show the output.

The output parameters of this command are the following:

{0} | {^wkname} Workspace name.

{1} | {^wkpath} Workspace path.

{2} | {^machine} Client machine name.

{3} | {^owner} Workspace owner.

{4} | {^guid} Workspace GUID.

{^tab} Inserts a tab space.

{^newline} Inserts a new line.

Examples:

cm ^gwp c:\myworkspace\code\file1.cpp --^format="Workspace name: {^wkname}"

== CMD\_DESCRIPTION\_HELP ==

Gets help for a Plastic SCM command.

== CMD\_USAGE\_HELP ==

Usage:

cm ^help <command>

== CMD\_HELP\_HELP ==

== CMD\_DESCRIPTION\_IOSTATS ==

Shows statistics about the hardware.

== CMD\_USAGE\_IOSTATS ==

Usage:

cm ^iostats [<repserverspec>] [<list\_of\_tests>[ ...]]

[--^nettotalmb=<value\_mb>] [--^networkiterations=<value\_iter>]

[--^diskdatasize=<value\_size>] [--^disktestpath=<value\_path>]

[--^systemdisplaytime=<value\_time>]

[--^systemdisplaytimeinterval=<value\_interval>]

Options:

repserverspec An available Plastic SCM server to perform

the network tests, such as "serverUploadTest"

and/ or "serverDownloadTest".

If no server is provided, the command tries

to communicate with the server configured by

default.

(Use 'cm ^help ^objectspec' to learn more about

server specs.)

list\_of\_tests Available tests. Use a whitespace to separate

test fields.

See Remarks for more info.

--^nettotalmb Indicates the amount of user data (in

MegaBytes) transmitted on a network test,

such as "^serverDownloadTest" or

"^serverUploadTest".

It must be a value between "4" and "512".

(Default: 16)

--^networkiterations Indicates the number of iterations of

"^serverDownloadTest" and/or "^serverUploadTest"

that will be run.

It must be a value between "1" and "100".

(Default: 1)

--^diskdatasize Indicates the amount of data (in MegaBytes)

that will be written and then read on the

"^diskTest".

It must be a value between "100" and "4096".

(Default: 512)

--^disktestpath Path where the "^diskTest" writes the test

files. If this parameter is not provided,

the command will try to use the system temp

path.

--^systemdisplaytime Time interval (in seconds) showing the usage

of system resources. This option is available

for the following tests: "^systemNetworkUsage"

and "^systemDiskUsage".

It must be a value between "1" and "3600".

(Default: 5 seconds).

--^systemdisplaytimeinterval Time interval (in seconds) between the

system performance samples. This option is

available for the following tests:

"^systemNetworkUsage" and "^systemDiskUsage".

It must be a value between "1" and "60".

(Default: 1 second).

== CMD\_HELP\_IOSTATS ==

Remarks:

This command requires an available server be used during the network

speed tests ("^serverUploadTest" and/or "^serverDownloadTest").

The '--^diskTestPath' must point to a path that belongs to the physical

disk drive about to be tested. If no path is specified, the command tries

to use the system default temp path.

The disk drive of the specified path must have enough free space to execute

the test.

During the command execution, the system can experience a degraded

performance caused by the tests performed.

Available tests:

--^serveruploadtest (Default) Measures the data upload speed from

Plastic SCM client to the server.

--^serverdownloadtest (Default) Measures the data download speed from

Plastic SCM server to the client.

--^disktest (Default) Measures the disk read speed and disk

write speed.

--^systemnetworkusage Shows the current usage of system network

resources.

(It shows Network Interface performance counters

provided by Microsoft Windows).

Available in Microsoft Windows only.

--^systemdiskusage Shows the current usage of system physical

disks.

(It shows Network Interface performance counters

provided by Microsoft Windows).

Available in Microsoft Windows only.

Examples:

cm ^iostats MYSERVER:8087 --^serveruploadtest --^serverdownloadtest --^nettotalmb=32

== CMD\_DESCRIPTION\_ISSUETRACKER ==

Gets, updates, or finds the issue status in the specified issue tracker.

== CMD\_USAGE\_ISSUETRACKER ==

Usage:

cm ^issuetracker <name> ^status ^get <task\_id> <parameter>[ ...]

cm ^issuetracker <name> ^status ^update <task\_id> <status> <parameter>[ ...]

cm ^issuetracker <name> ^status ^find <status> <parameter>[ ...]

cm ^issuetracker <name> ^connection ^check <parameter>[ ...]

name Name of the issue tracker to connect with.

Only Jira is supported at the moment.

task\_id Number of the issue to query or update.

^status A valid status for an issue in the issue tracker.

Jira parameters (all are mandatory):

--^user=<user> The user to authenticate.

--^password=<password> The password to authenticate.

--^host=<url> The target url of the issue tracker.

--^projectkey=<key> The project key of Jira project.

== CMD\_HELP\_ISSUETRACKER ==

Examples:

cm ^issuetracker jira ^status ^get 11 --^user=user@mail.es --^password=pwd \

--^host=https://user.atlassian.net --^projectkey=PRJ

(Gets the status of the issue 11 for the 'PRJ' project.)

cm ^issuetracker jira ^status ^update 11 "Done" --^user=user@mail.es \

--^password=pwd --^host=https://user.atlassian.net --^projectkey=PRJ

(Updates the status to 'Done' of the issue 11 for the 'PRJ' project.)

cm ^issuetracker jira ^status ^find "Done" --^user=user@mail.es --^password=pwd \

--^host=https://user.atlassian.net --^projectkey=PRJ

(Gets the task ids whose status is set to 'Done' for the 'PRJ' project-)

cm ^issuetracker jira ^connection ^check --^user=user@mail.es --^password=pwd \

--^host=https://user.atlassian.net --^projectkey=PRJ

(Checks whether the configuration parameters are valid or not.)

== CMD\_DESCRIPTION\_LICENSEINFO ==

Displays license information and license usage.

== CMD\_USAGE\_LICENSEINFO ==

Usage:

cm ^licenseinfo | ^li [--^server=<repserverspec>] [--^inactive] [--^active]

[--^sort=(^name|^status)]

Options:

--^server Gets the license info from the specified server.

If no server is specified, executes the command on the

one configured on the client.

(Use 'cm ^help ^objectspec' to learn more about repserver

specs.)

--^inactive Shows only inactive users in the "license usage" section.

--^active Shows only active users in the "license usage" section.

--^sort Sorts users by one of the specified sort options:

'^name' or '^status'.

== CMD\_HELP\_LICENSEINFO ==

Remarks:

The information displayed consists of expiration date, activated and

deactivated users, etc.

Examples:

cm ^licenseinfo

cm ^licenseinfo --^server=myserver:8084

cm ^licenseinfo --^sort=^name

== CMD\_DESCRIPTION\_LINKTASK ==

Links a changeset to a task.

== CMD\_USAGE\_LINKTASK ==

This is an automation command, meant to be used to automate 'cm' only.

It is not as user friendly as it should be.

Usage:

cm ^linktask | ^lt <csetspec> <ext\_prefix> <task\_name>

csetspec The full changeset specification to link to a task.

(Use 'cm ^help ^objectspec' to learn more about changeset

specs.)

ext\_prefix The extension prefix of the configured issue tracking

system to work with.

task\_name The task identifier on the issue tracking system.

== CMD\_HELP\_LINKTASK ==

Examples:

cm ^lt ^cs:8@^rep:default@^repserver:localhost:8084 jira PRJ-1

== CMD\_DESCRIPTION\_LOCK\_LIST ==

Shows locks on a server.

== CMD\_USAGE\_LOCK\_LIST ==

Usage:

cm ^lock ^list | ^ls [<revspec> [ ...]] [--^server=<server>]

[--^onlycurrentuser] [--^onlycurrentworkspace]

[--^ignorecase]

revspec If one or more are present, this command will display

one lock line for each specified revision if its

associated item is locked in the server. Otherwise,

this command will list all locked items in the default

server (or the one set with the '--^server' option)

Use a whitespace to separate the rev specs when using

more than one.

(Use 'cm ^help ^objectspec' to learn more about rev specs.)

Options:

--^server Repository server specification.

This option will override the default server which

is retrieved from the current workspace or the

client.conf file.

(Use 'cm ^help ^objectspec' to learn more about

server specs.)

--^onlycurrentuser Filters the results showing only the locks performed

by the current user.

--^onlycurrentworkspace Filters the results showing only the locks performed

on the current workspace (matching them by name).

--^ignorecase Ignores casing on the paths when a serverpath spec

is used. With this flag, the command will work for

"/src/foo.c" even if the user writes "/sRc/fOO.c".

== CMD\_HELP\_LOCK\_LIST ==

Remarks:

The command will display a list of the currently locked items in the

default server. It also accepts a list of revision specifications; in that

case, only the locks belonging to the selected items will be displayed.

A '--^server=<server>' can be used to set the default server to query.

The command shows a line for every lock in the specified server:

- GUID of the locked item.

- User name who performed the lock.

- Workspace name where the lock was performed.

- Path of the locked item (server path format).

Examples:

cm ^lock ^list

cm ^lock ^list --^server=myserver:8084

cm ^lock ^ls ^serverpath:/src/foo.c#^cs:99@default@localhost:8084

cm ^lock ^list ^revid:3521@default ^itemid:2381@secondary --^onlycurrentuser

cm ^lock ^ls --^onlycurrentuser

cm ^lock ^ls --^onlycurrentuser --^onlycurrentworkspace

== CMD\_DESCRIPTION\_LISTUSERS ==

Lists users and groups.

== CMD\_USAGE\_LISTUSERS ==

Usage:

cm ^listusers | ^lu <repserverspec> [--^onlyusers] [--^onlygroups]

[--^filter= <str\_filter>]

repserverspec Repository server specification.

(Use 'cm ^help ^objectspec' to learn more about specs.)

Options:

--^onlyusers Lists only users.

--^onlygroups Lists only groups.

--^filter Lists only users and/or groups that matches the

specified filter.

== CMD\_HELP\_LISTUSERS ==

Examples:

cm ^lu localhost:8084

(Lists all users in the server.)

cm ^listusers localhost:8084 --^onlyusers --^filter=m

(Lists only the users in the server that contains "m".)

== CMD\_DESCRIPTION\_LOCATION ==

Returns the path of 'cm'.

== CMD\_USAGE\_LOCATION ==

Usage:

cm ^location

== CMD\_HELP\_LOCATION ==

== CMD\_DESCRIPTION\_LOCK ==

This command allows the user to manage locks.

== CMD\_USAGE\_LOCK ==

Usage:

cm ^lock <command> [options]

Commands:

^list | ^ls

^unlock

To get more information about each command run:

cm ^lock <command> --^usage

cm ^lock <command> --^help

== CMD\_HELP\_LOCK ==

Examples:

cm ^lock ^list

cm ^lock

('^list' is optional if there are no arguments.)

cm ^lock ^ls ^serverpath:/src/foo.c#^cs:99@default@localhost:8084

cm ^lock ^unlock 91961b14-3dfe-4062-8c4c-f33a81d201f5

== CMD\_DESCRIPTION\_LOG ==

Gets info about revisions in changesets.

== CMD\_USAGE\_LOG ==

Usage:

cm ^log [<csetspec> | <repspec>] [--^from=<csetspec\_from>] [--^allbranches]

[--^ancestors] [--^csformat=<str\_format>] [--^itemformat=<str\_format>]

[--^xml[=<output\_file>]] [--^encoding=<name>]

[--^repositorypaths | --^fullpaths | --^fp]

Options:

csetspec The command will return every change made in the

changeset which specification is provided.

(Use 'cm ^help ^objectspec' to learn more about changeset

specs.)

repspec The command will list all changes made in the specified

repository.

(Use 'cm ^help ^objectspec' to learn more about repository

specs.)

--^from Lists all the changes made in every changeset from the

changeset specification [csetspec\_from] to the

changeset specification [csetspec].

The [csetspec\_from] changeset is not included in the

output.

Ignored when a repository spec is provided.

--^allbranches Shows information about the changesets created in a

specified interval, for all the branches where those

changesets were created.

--^ancestors Shows information about the reachable changesets by

following the parent and merge links for the given

changeset ([csetspec]). If the from changeset

([csetspec\_from]) is provided too, it will be used as

lower limit for all the paths. Remarks: The changeset

changes will not be shown when this option is used.

--^csformat Retrieves the changeset info in a specific format. See

Remarks for more info.

--^itemformat Retrieves the item info in a specific format. See

Remarks for more info.

--^xml Prints the output in XML format to the standard output.

It is possible to specify an output file.

--^encoding Used with the '--^xml' option, specifies the encoding to

use in the XML output, i.e.: utf-8.

See the MSDN documentation at

http://msdn.microsoft.com/en-us/library/system.text.encoding.aspx

to get the table of supported encodings and its format,

(at the end of the page, in the "Name" column).

--^fullpaths, --^fp Force printing full workspace paths for files and

directories when possible.

--^repositorypaths Prints repository paths (server paths) instead of

workspace paths. (This option overrides the

'--^fullpaths' option).

== CMD\_HELP\_LOG ==

Remarks:

- If neither 'csetspec' nor option is specified, the command shows

information about every changeset created within the last month in every

branch.

- If only the option '--^from' is included, the command shows the

information about every changeset from that specified changeset to the

last changeset, in the branch where the changeset was created.

- If the option '--^allbranches' appears without an interval, the command

retrieves the same information as it would do if only 'csetspec' was

specified.

- If the '--^from' is used, the output contains information from the

'csetspec\_from'+1 on.

- The repository used to show the changeset information is the one loaded

in the path where the command executes on.

This command accepts a format string for the items ('--^itemformat') and a

format string for the changesets ('--^csformat').

The output parameters of '--^csformat' are the following:

{^tab} Inserts a tab space.

{^newline} Inserts a new line.

{^changesetid} Changeset number.

{^branch} Branch where the changeset was created.

{^date} Date of the changeset.

{^owner} Owner of the changeset.

{^comment} Comment of the changeset.

{^items} Items involved in the changeset.

{^repository} Repository where the changeset exists.

{^repserver} Server name.

The output parameters of '--^itemformat' are the following:

{^tab} Inserts a tab space.

{^newline} Inserts a new line.

{^path} Item path.

{^branch} Branch where the changeset was created.

{^date} Date of the changeset.

{^owner} Owner of the changeset.

{^shortstatus} Prints the short format. See below.

{^fullstatus} Prints the long format. See below.

Short format and its corresponding long format:

'^A' ^Added

'^D' ^Deleted

'^M' ^Moved

'^C' ^Changed

These are valid output strings:

--^csformat="{^newline}Changeset {^changesetid} created on {^date};{^tab} changed items: {^items}."

--^itemformat="{^newline}The item {^path} was changed in the branch {^branch}."

Examples:

cm ^log

(Shows information about every changeset created in the last month in every

branch.)

cm ^log ^cs:16

(Shows information about the changes done in the changeset 16 in the branch

where the changeset was created.)

cm ^log ^cs:16 --^csformat="{^newline}Changeset {^changesetid} created on \

{^date};{^tab} changed items: {^items}."

(Shows the information in the specified format.)

cm ^log --^from=^cs:20 ^cs:50

(Shows the information about every revision contained in every changeset

from the changeset 21 to the changeset 50.)

cm ^log --^from=^cs:20 ^cs:50 --^allbranches

(Shows the information about every revision contained in every changeset

from the changeset 21 to the changeset 50 in every branch of the

repository.)

cm ^log ^rep:myrep@localhost:8084

(Shows information about the changes done in the specified repository.

No workspace is required to run the command.)

cm ^log --^from=^cs:20@^rep:mainRep@localhost:8084

(Shows the information about every revision contained in every changeset

from the changeset 21. No workspace is required to run the command, because

the full changeset spec was specified.)

== CMD\_DESCRIPTION\_LS ==

Lists the contents of a tree.

== CMD\_USAGE\_LS ==

Usage:

cm ^ls | ^dir [<paths>[ ...]] [--^format=<str\_format>] [--^symlink]

[--^selector[=<selector\_format>]] [--^tree=<obj\_spec>]

[-^R | -^r | --^recursive]

[--^xml[=<output\_file>]] [--^encoding=<name>]

Options:

paths List of paths to show. Use a whitespace to separate

paths.

Use double quotes (" ") to specify paths containing

spaces.

--^format Retrieves the output message in a specific format. See

Remarks for more info.

--^symlink Applies the operation to the symlink and not to the

target.

--^selector Gets the content from the active workspace selector.

If selector\_format is specified, then lists the

specified selector.

This is mostly deprecated since selectors are

no longer a central part of Plastic SCM since 4.x.

--^tree Lists the tree in the specified changeset or branch.

(Use 'cm ^help ^objectspec' to learn more about specs.)

-^R Lists recursively.

--^xml Prints the output in XML format to the standard output.

It is possible to specify an output file.

--^encoding Used with the '--^xml' option, specifies the encoding to

use in the XML output, i.e.: utf-8.

See the MSDN documentation at

http://msdn.microsoft.com/en-us/library/system.text.encoding.aspx

to get the table of supported encodings and its format,

(at the end of the page, in the "Name" column).

== CMD\_HELP\_LS ==

Remarks:

- Path can be typed with meta-characters.

- The list depends on the workspace selector.

- The output of the command can be formatted specifying a format string.

- If '--^tree' or '--^selector' options are specified, the given

path must be a server path (a.k.a.: 'cm path'): /dir/file.txt, not a

workspace path: C:\Users\myuser\mywk\dir\file.txt

- If no path is provided, the workspace path assumed is the current

directory. If '--^tree' or '--^selector' options are used, then

the root path ("/") is assumed.

The default format string is:

"{^size,10} {^date:dd/MM/yyyy} {^date:HH:mm}\

{^type,-6} {^location,-12} {^checkout,-5} {^name}\

{^symlinktarget}"

This command accepts a format string to show the output.

The output parameters of this command are the following:

{^size}

{^formattedsize}

{^date}

{^type}

^dir directory,

^txt text file,

^File file.

{^location} Example: ^br:branch#cset

{^checkout}

{^name}

{^changeset}

{^path}

{^repspec}

{^owner}

{^revid}

{^parentrevid}

{^itemid}

{^brid}

{^repid}

{^server}

{^symlinktarget}

{^hash}

{^chmod}

{^wkpath} Path relative to workspace root

{^branch}

{^newlocation} cset@branch

{^guid} (Will take longer to resolve)

{^itemguid}

{^transformed} Show applied rule for transformed items

You can customize the '^ls' format setting the PLASTIC\_LS\_FORMAT environment

variable.

Examples:

cm ^ls

cm ^ls c:\workspace\src

cm ^ls --^format={^name}

(Only file names.)

cm ^ls --^symlink

(Displays information about the symlink instead of the 'symlinked' file or

directory. Available on UNIX environments.)

cm ^ls code --^selector

(Shows the content of the 'code' subdirectory from the current workspace

selector.)

cm ^ls /code --^selector="^rep 'myrep' ^path '/' ^branch '/^main'"

(Shows the content of the '/code' subdirectory on the specified selector.

Note that the path is specified in server format.)

cm ^ls /code --^tree=44@myrep@denver:7070

(Lists the '/code' subdirectory at changeset 44 at repo 'myrep' at server

'denver:7070'.)

cm ^ls /code --^tree=^br:/main/scm13596@myrep@denver:7070

(Lists the '/code' subdirectory at the latest changeset in branch

'/main/scm13596' at repo 'myrep' at server 'denver:7070'.)

cm ^ls /code --^tree=ae1390ed-7ce9-4ec3-a155-e5a61de0dc77@myrep@denver:7070

(Lists the '/code' subdirectory at changeset

ae1390ed-7ce9-4ec3-a155-e5a61de0dc77 at repo 'myrep' at server

'denver:7070'.)

== CMD\_DESCRIPTION\_TRIGGER\_LIST ==

Lists the triggers of a given type on a server.

== CMD\_USAGE\_TRIGGER\_LIST ==

Usage:

cm ^trigger | ^tr ^list | ^ls [<subtype-type>] [--^server=<repserverspec>]

[--^format=<str\_format>]

Options:

subtype-type Trigger execution and trigger operation.

Type 'cm ^showtriggertypes' to see a list of trigger

types.

--^server Lists the triggers on the specified server.

If no server is specified, executes the command on the

one configured on the client.

(Use 'cm ^help ^objectspec' to learn more about server

specs.)

--^format Retrieves the output message in a specific format. See

Remarks for more info.

== CMD\_HELP\_TRIGGER\_LIST ==

Remarks:

If the type is not specified, lists all the triggers on the server.

This command accepts a format string to show the output.

The output parameters of this command are the following:

{0} Trigger position.

{1} Trigger name.

{2} Trigger path.

{3} Trigger owner.

{4} Trigger type.

{5} Trigger filter.

Examples:

cm ^trigger list after-mklabel

cm ^tr ^ls ^before-mkbranch --^server=myserver:8084

== CMD\_DESCRIPTION\_MANIPULATESELECTOR ==

Changes the selector to a date.

== CMD\_USAGE\_MANIPULATESELECTOR ==

This is an automation command, meant to be used to automate 'cm' only.

It is not as user friendly as it should be.

Usage:

cm ^manipulateselector | ^ms [<wk\_path> | <wk\_spec>] --^atdate=<sel\_date>

wk\_path Path of the workspace.

wk\_spec Workspace specification. (Use 'cm ^help ^objectspec' to

learn more about specs.)

Options:

--^atdate Returns a selector that will recreate the workspace as

it would have looked at the specified date.

== CMD\_HELP\_MANIPULATESELECTOR ==

Remarks:

If neither path nor workspace spec is specified, the command will take the

current directory as workspace path.

Examples:

cm ^manipulateselector c:\workspace --^atdate=yyyy-MM-ddTHH:mm:ss

cm ^manipulateselector --^atdate=yyyy-MM-ddTHH:mm:ss

cm ^manipulateselector > mySelector.txt --^atdate=yyyy-MM-ddTHH:mm:ss

cm ^manipulateselector ^wk:build\_wk@BUILDER --^atdate=yyyy-MM-ddTHH:mm:ss

== CMD\_DESCRIPTION\_MERGE ==

Merges a branch with another branch.

== CMD\_USAGE\_MERGE ==

Usage:

cm ^merge <source\_spec> [--^merge] [--^cherrypicking] [--^forced]

[--^mergetype=(^onlyone|^onlysrc|^onlydst|^try|^forced)]

[--^interval-origin=<csetspec> | --^ancestor=<csetspec>]

[--^keepsource | --^ks] [--^keepdestination | --^kd]

[--^automaticresolution=<conflict-types>[;...]]

[--^subtractive] [--^mount] [--^printcontributors]

[--^noprintoperations] [--^silent]

[(--^to=<brspec> | --^destination=<brspec>)[--^shelve]]

[--^no-dst-changes]

[-^c=<str\_comment> | --^commentsfile=<comments\_file>]

[--^resolveconflict --^conflict=<index>

--^resolutionoption=(^src|^dst|(^rename --^resolutioninfo=<strname>))

--^mergeresultfile=<path> --^solvedconflictsfile=<path>]

[--^nointeractiveresolution]

[--^machinereadable [--^startlineseparator=<sep>]

[--^endlineseparator=<sep>] [--^fieldseparator=<sep>]]

source\_spec Specification of the source object to merge from:

- a branch specification: '[^br:/]br\_name'

- a label specification: '^lb:lb\_name'

- a changeset specification: '^cs:cs\_number'

- a shelve specification: '^sh:shelve\_number'

(Use 'cm ^help ^objectspec' to learn more about specs.)

Options:

--^merge Performs the merge. Otherwise, prints the

conflicts found.

--^cherrypicking Merges the changes included on the source

changesets. This option is not used if the merge

source specification is a label.

--^forced Does not check if the source and destination are

already connected.

This option is only available for interval merge

and cherrypicking.

--^mergetype See Remarks for more info.

--^interval-origin Specifies which changeset is chosen as the

interval origin, so the merge will only take the

differences between the source changeset and the

specified interval origin.

--^ancestor This is an alias of --^interval-origin.

--^keepsource Accepts all changes from source contributor for

items with conflicts.

--^keepdestination Preserves changes from destination contributor

for items with conflicts.

--^automaticresolution Used to resolve directory conflicts. This option

lets you choose whether the source or the

destination contributor should be automatically

selected to resolve the conflict.

Use a semicolon to separate conflict types.

See Remarks for more info.

--^subtractive Deletes changes introduced by a merge. The

parameter passed to the command (source\_spec) is

used to specify which is the source to delete

changes. It must be a changeset. In the case of a

changeset interval, the '--^interval-origin' must

be used to define the interval origin. To remove

a change, the system creates a new checked out

revision which will have the content of the

previous one except for the deleted changes.

--^mount The mount point for the given repository.

--^printcontributors Prints the contributors (base, source, and

destination).

--^noprintoperations Silently resolves merges without showing

information about the resolution.

--^silent Does not show any output.

--^to | --^destination Performs a merge-to operation to the specified

branch (by entering a branch spec or brspec)

with full conflict resolution.

A "merge-to" (or workspace-less merge) is a merge

done in the server side. While normal merges

happen on a workspace merging "from" a branch,

label or changeset, a merge-to happens entirely

on the server. While in normal merges the

"destination" is the workspace, in "merge-to" a

destination must be always specified (that's why

we call it "to").

Check the following link for more information

about the "merge to" feature:

https://www.plasticscm.com/download/help/mergeto

--^shelve Creates a shelve with the changes of the merge

result (plus merge traceability info) instead of

creating a new changeset. This option is not

available when the merge source is a shelve. This

option is only available for server-side-merge

(a.k.a. "merge-to"). Hence, the '--^to' and

'--^merge' options are required.

--^no-dst-changes Ensures that the destination contributor doesn't

have changes (the destination changeset is also

the common ancestor). When there are changes on

the destination, the merge is not allowed.

-^c Applies the specified comment to the changeset

created in the merge operation.

--^commentsfile Applies the comment in the specified file to the

changeset created in the merge operation.

--^resolveconflict (Mainly used by plugins. See Remarks for more info.)

Used to solve a directory conflict.

--^conflict Used with the '--^resolveconflict' flag, specifies

the index of the conflict to solve starting at 1.

--^resolutionoption Used with the '--^resolveconflict' flag, indicates

the type of the conflict resolution. Use one of

the following options: '^src', '^dst', '^rename'.

See Remarks for more info.

--^resolutioninfo Used with the '--^resolveconflict' flag, provides

the name to use when the '--^resolutionoption'

option is 'rename'.

--^mergeresultfile Used with the '--^resolveconflict' flag, outputs

into a file the information of the merge result

between different calls .The specified path will

be created during the first call and updated on

each next call.

--^solvedconflictsfile Used with the '--^resolveconflict' flag, outputs

into a file the information of the conflicts

solved between different calls. The specified

path will be created during the first call and

updated on each next call.

--^nointeractiveresolution (Mainly used by plugins. See Remarks for more info.)

Avoids prompting the user for manual conflict.

This way, a directory conflict won't be solved.

--^machinereadable (Mainly used by plugins. See Remarks for more info.)

Outputs the result in an easy-to-parse format.

--^startlineseparator Used with the '--^machinereadable' flag, specifies

how the lines should start. (Default: empty string.)

--^endlineseparator Used with the '--^machinereadable' flag, specifies

how the lines should end. (Default: empty string.)

--^fieldseparator Used with the '--^machinereadable' flag, specifies

how the fields should be separated. (Default:

whitespace.)

== CMD\_HELP\_MERGE ==

Remarks:

This command is used to merge changes between two branches or between a

label and a branch. The destination of the merge must be always a branch.

The merge source is specified as an argument.

Destination is the current content of the workspace.

For example, to display the elements that will be merged from branch

task001 to the main branch, the selector must point to the main branch,

the workspace must be updated, and then:

cm ^merge ^br:/task001

To really perform the merge, '--^merge' option is added:

cm ^merge ^br:/task001 --^merge

To define the merge source, the following specs can be used:

- A branch specification (brspec):

[^br:/]br\_name

Example: ^br:/main/task001

(The above performs a merge from the last changeset on this branch.)

- A label specification (lbspec):

^lb:lb\_name

Example: ^lb:BL001

(Merge from the labeled changeset.)

- A changeset specification (csetspec):

^cs:cs\_number

Example: ^cs:25

(Merge from the given changeset content.)

- A shelve specification (shspec):

^sh:shelve\_number

Example: ^sh:2

(Merge from the given shelve content.)

To automatically resolve directory conflicts, use '--^automaticresolution'

option and specify the type of conflict followed by the contributor

(source or destination) that must be selected during the merge operation.

(Separate each "type of conflict"-"contributor" pair by a semicolon (;).)

For example:

cm ^merge ^cs:2634 --^merge --^automaticresolution=^eviltwin-src;^changedelete-src

(The merge operation from changeset 2634 resolves the "^eviltwin" and

"^changedelete" conflicts by keeping the source ("-^src") contributor in

both cases.)

- A "-^src" suffix after a conflict type tells the merge command to keep the

source contributor changes.

- A "-^dst" suffix will keep the destination contributor changes.

This is the list of conflict types the merge command supports:

"^movedeviltwin", "^eviltwin", "^changedelete", "^deletechange", "^movedelete",

"^deletemove", "^loadedtwice", "^addmove", "^moveadd", "^divergentmove",

"^cyclemove", "^all".

The "^all" value overrides the other options. In the following example,

"^eviltwin-dst" will be ignored:

cm ^merge ^br:/main/task062 --^merge --^automaticresolution=^all-src;^eviltwin-dst

Check the following link to learn more about merge conflicts:

https://www.plasticscm.com/download/help/directorymerges

These are the options for '--^mergetype':

^onlyone Automatic merge if only one contributor modified the

item.

^onlysrc Automatic merge if only source contributor modified the

item.

^onlydst Automatic merge if only destination contributor modified

the item.

^try Automatic merge if only one contributor has modified the

conflictive piece of code (each conflict).

^forced Always try to solve all non-automatic conflicts.

These are the options that are mainly used by plugins and integrations:

- '--^resolveconflict' to solve a directory conflict. You also have to

use the following options:

- '--^conflict' is the index of the conflict that you want to

solve, starting at 1.

- '--^resolutionoption' indicates the conflict resolution to

use. This can be:

- '^src' to keep the source change and discard the

destination change

- '^dst' to keep the destination change and discard the

source change

- '^rename' (only if the conflict type supports this

resolution), to rename the destination to the given name

provided with the '--^resolutioninfo' option.

- '--^resolutioninfo' to provide the name to use on a

'^rename' resolution

- '--^mergeresultfile' and '--^solvedconflictsfile', both used to

store the merge info between different calls.

- '--^nointeractiveresolution' indicates the merge to not ask the user for

manual conflict resolution.

- '--^machinereadable' and '--^startlineseparator', '--^endlineseparator',

'--^fieldseparator' options to print the output on a machine-readable

way (easier-to-parse).

Example:

cm ^merge --^machinereadable --^startlineseparator=start@\_@line \

--^endlineseparator=new@\_@line --^fieldseparator=def#\_#sep \

--^mergeresultfile=C:\Users\Borja\AppData\Local\Temp\2tmp4D6C.tmp \

--^solvedconflictsfile=C:\Users\Borja\AppData\Local\Temp\2tmp4D6D.tmp \

--^resolveconflict --^conflict=1 --^resolutionoption=rename \

--^resolutioninfo=bin\_dst ^br:/main/task --^merge

Examples:

cm ^merge ^br:/task001

(Does not merge, just prints items to be merged.)

cm ^merge ^br:/task001 --^merge

(Does merge from branch 'task001'.)

cm ^merge ^cs:5 --^merge --^cherrypicking --^interval-origin=^cs:2

(Cherrypick from the changeset interval (2,5].)

cm ^merge ^cs:8 --^merge --^subtractive --^keepdestination

(Subtractive merge from changeset 8, keeping destination changes for those

elements with conflicts.)

cm ^merge ^br:/main/task001 --^to=^br:/main --^merge -^c="Integrated new UI"

(Does server-side merge, a.k.a. merge-to, from branch 'task001' to branch

'main' and sets a comment.)

cm ^merge ^br:/main/task001 --^to=^br:/main --^merge --^shelve

(Does server-side merge from branch 'task001' to branch 'main' and leaves

the result on a shelve.)

cm ^merge ^sh:2 --^to=^br:/main --^merge --^no-dst-changes

(Applies the shelve 2 into 'main' only if it was created from the current

'main' head')

== CMD\_DESCRIPTION\_ATTRIBUTE\_CREATE ==

Creates a new attribute.

== CMD\_USAGE\_ATTRIBUTE\_CREATE ==

Usage:

cm ^attribute | ^att ^create | ^mk <att\_name>

att\_name Attribute name

== CMD\_HELP\_ATTRIBUTE\_CREATE ==

Examples:

cm ^attribute ^create status

(Creates the attribute 'status'.)

cm ^att ^mk integrated

(Creates the attribute 'integrated'.)

== CMD\_DESCRIPTION\_BRANCH ==

Allows the user to manage branches.

== CMD\_USAGE\_BRANCH ==

Usage:

cm ^branch | ^br <command> [options]

Commands:

^create | ^mk

^delete | ^rm

^rename

^history

^showmain

^showmerges

To get more information about each command run:

cm ^branch <command> --^usage

cm ^branch <command> --^help

== CMD\_HELP\_BRANCH ==

Examples:

cm ^branch /main/scm21345

cm ^branch ^create /main/scm21345

cm ^branch ^delete /main/scm21345

cm ^branch ^rename /main/scm21345 scm21346

cm ^branch ^history /main/scm21345

cm ^branch ^showmain

cm ^branch ^showmerges file.txt

== CMD\_DESCRIPTION\_BRANCH\_CREATE ==

Creates a new branch.

== CMD\_USAGE\_BRANCH\_CREATE ==

Usage:

cm ^branch | ^br [^create | ^mk] <brspec>

[--^changeset=<csetspec> | --^label=<lbspec>]

[-^c=<str\_comment> | -^commentsfile=<comments\_file>]

brspec The new branch name or spec.

(Use 'cm ^help ^objectspec' to learn more about branch specs.)

Options:

--^changeset Changeset used as starting point for the new branch.

(Use 'cm ^help ^objectspec' to learn more about cset specs.)

--^label Label used as starting point for the new branch.

(Use 'cm ^help ^objectspec' to learn more about label specs.)

-^c Fills in the comment field of the new branch with the

specified text.

-^commentsfile Fills in the comment field of the new branch with the

contents of the specified file.

== CMD\_HELP\_BRANCH\_CREATE ==

Remarks:

To create a top-level branch, specify the name without any hierarchy.

For example:

cm ^br /dev

If no optional parameter '--^changeset' is specified, the base of the new

branch will be the last changeset on the parent branch. If the new branch

is a top-level branch, the base changeset used will be cset 0.

You can specify a comment using either the '-^c' or the '-^m' switches:

cm ^branch /main/task001 -^c="This is the comment"

cm ^branch /main/task001 -^m "This is the comment"

Set the PLASTICEDITOR environment variable to specify an editor for

entering comments. If the PLASTICEDITOR environment variable is set, and

the comment is empty, the editor will be automatically launched to allow

you to specify the comment.

Examples:

cm ^branch task001

cm ^branch ^create task001

cm ^branch ^mk task001

cm ^br ^mk task001

(Creates a top-level 'task001' branch in the repository of the current

workspace.)

cm ^branch ^br:/task001/task002@

(Creates 'task002' branch as child of 'task001'.)

cm ^br /main/task001@myrep@myserver:8084 -^c="my comment"

(Creates 'task001' branch as child of 'main' in repository

'myrep@myserver:8084' with comment 'my comment'.)

cm ^branch ^br:/main/task001 --^changeset=2837 -^commentsfile=commenttask001.txt

(Creates the 'task001' branch as child of 'main' with base 'changeset=2837',

and applies the comment in 'commenttask001.txt' file.)

== CMD\_DESCRIPTION\_BRANCH\_DELETE ==

Deletes one or more branches.

== CMD\_USAGE\_BRANCH\_DELETE ==

Usage:

cm ^branch | ^br ^delete | ^rm <brspec>[ ...]

brspec Branch to delete. Use a whitespace to separate branches.

(Use 'cm ^help ^objectspec' to learn more about branch

specs.)

== CMD\_HELP\_BRANCH\_DELETE ==

Remarks:

This command deletes one or more branches.

Examples:

cm ^branch ^delete /main/task001

(Deletes the branch with name 'task001' that is a child of 'main' in the

repository of the current workspace.)

cm ^br ^rm main/task002 /main/task012@reptest@myserver:8084

(Deletes branches '/main/task002' in the repository of the current workspace

and '/main/task012' in the repository 'reptest@myserver:8084'.)

== CMD\_DESCRIPTION\_BRANCH\_RENAME ==

Renames a branch.

== CMD\_USAGE\_BRANCH\_RENAME ==

Usage:

cm ^branch | ^br ^rename <brspec> <new\_name>

brspec Branch to rename.

(Use 'cm ^help ^objectspec' to learn more about branch specs.)

new\_name New name for the branch.

== CMD\_HELP\_BRANCH\_RENAME ==

Remarks:

This command renames a branch.

Examples:

cm ^branch ^rename /main/task0 task1

(Renames branch '/main/task0' to '/main/task1'.)

cm ^br ^rename ^br:/main@reptest@server2:8084 secondary

(Renames the 'main' branch of repository 'reptest' to 'secondary'.)

== CMD\_DESCRIPTION\_BRANCH\_HISTORY ==

Shows the history of a branch.

== CMD\_USAGE\_BRANCH\_HISTORY ==

Usage:

cm ^branch | ^br ^history <brspec> [--^dateformat=<date\_format>]

[--^machinereadable]

brspec The branch specification to obtain the history.

(Use 'cm ^help ^objectspec' to learn more about branch specs.)

Options:

--^dateformat Format used to output dates.

--^machinereadable Outputs the result in an easy-to-parse format.

== CMD\_HELP\_BRANCH\_HISTORY ==

Examples:

cm ^branch ^history ^br:/main/scm001@myrepository@myserver:8084

(Displays the history of '/main/scm001' branch of 'myrepository' repository

on 'myserver' server.)

cm ^br ^history main --^dateformat="yyyy, dd MMMM" --^machinereadable

(Displays the history of the 'main' branch of the current repository,

with a given date format, and in an easy-to-parse format.)

== CMD\_DESCRIPTION\_BRANCH\_SHOWMAIN ==

Shows the main branch of a repository.

This is an automation command, meant to be used to automate 'cm' only.

Most likely, the main branch of your repository is '/main'.

== CMD\_USAGE\_BRANCH\_SHOWMAIN ==

Usage:

cm ^branch | ^br ^showmain [<repspec>] [--^encoding=<name>]

[--^format=<format\_str>] [--^dateformat=<date\_format>]

repspec The repository specification where to show the main

branch.

(Use 'cm ^help ^objectspec' to learn more about rep specs.)

Options:

--^encoding Specifies the encoding to use in the output,

i.e.: utf-8.

See the MSDN documentation at

http://msdn.microsoft.com/en-us/library/system.text.encoding.aspx

to get the table of supported encodings and its format,

(at the end of the page, in the "Name" column).

--^format Retrieves the output message in a specific format. See

Remarks for more info.

--^dateformat Format used to output dates.

== CMD\_HELP\_BRANCH\_SHOWMAIN ==

Remarks:

This command shows the main branch of a repository.

This command accepts a format string to show the output.

The output parameters of this command are the following:

{^id} Branch id.

{^comment} Comment.

{^date} Date.

{^name} Name.

{^owner} Owner.

{^parent} Parent branch name.

{^repository} Repository.

{^repname} Repository name.

{^repserver} Server.

{^changeset} Head changeset of the branch.

Examples:

cm ^branch ^showmain

(Displays the main branch for the repository of the current workspace.)

cm ^branch ^showmain repo@server:8084

(Displays the main branch for the repository 'repo' in server

'server:8084'.)

cm ^br ^showmain --^dateformat="yyyy, dd MMMM" --^encoding=utf8

(Displays the main branch of the repository with a given date format,

and the output is in utf8.)

cm ^br ^showmain --^format="{^id} - {^name}"

(Displays the main branch of the repository, printing only its id and name.)

== CMD\_DESCRIPTION\_BRANCH\_SHOWMERGES ==

Shows branches pending to be merged.

== CMD\_USAGE\_BRANCH\_SHOWMERGES ==

This is an automation command, meant to be used to automate 'cm' only.

It is not as user friendly as it should be.

Usage:

cm ^branch | ^br ^showmerges <item\_path>[ ...]

[--^format=<format\_str>]

[--^dateformat=<date\_format>]

Options:

--^format Retrieves the output message in a specific format. See

Remarks for more info.

--^dateformat Format used to output dates.

== CMD\_HELP\_BRANCH\_SHOWMERGES ==

Remarks:

This command accepts a format string to show the output.

The output parameters of this command are the following:

{^id} Branch id.

{^comment} Comment.

{^date} Date.

{^name} Name.

{^owner} Owner.

{^parent} Parent branch name.

{^parentid} Parent branch id.

{^repid} Repository id.

{^repository} Repository.

{^repname} Repository name.

{^repserver} Repository server.

Examples:

cm ^branch ^showmerges file.txt

(Displays branches involved in the pending merge of 'file.txt'.)

cm ^branch ^showmerges file.txt --^format="{^date} {^name}" --^dateformat="ddMMyy"

(Displays branches involved in the merge, printing only the date and the

name, with a given date format.)

== CMD\_DESCRIPTION\_REPOSITORY ==

Allows the user to manage repositories.

== CMD\_USAGE\_REPOSITORY ==

Usage:

cm ^repository | ^repo <command> [options]

Commands:

^create | ^mk

^delete | ^rm

^list | ^ls

^rename

^add

To get more information about each command run:

cm ^repository <command> --^usage

cm ^repository <command> --^help

== CMD\_HELP\_REPOSITORY ==

Examples:

cm ^repository

cm ^repository ^list

cm ^repository newrepo

cm ^repository ^create newrepo

cm ^repository ^rename oldname newname

cm ^repository ^add C:\repo\

== CMD\_DESCRIPTION\_REPOSITORY\_CREATE ==

Creates a repository on a server.

== CMD\_USAGE\_REPOSITORY\_CREATE ==

Usage:

cm ^repository | ^repo <rep\_name>

cm ^repository | ^repo <repserverspec> <rep\_name>[ ...]

cm ^repository | ^repo [^create | ^mk] <rep\_name>

repserverspec Repository server specification.

(Use 'cm ^help ^objectspec' to learn more about rep server

specs.)

rep\_name Name or names of the new repository or repositories.

Use a whitespace to separate repository names.

== CMD\_HELP\_REPOSITORY\_CREATE ==

Examples:

cm ^repository MyRep

cm ^repo 192.168.1.140:8087 Rep01 Rep01/ModuleA Rep01/ModuleB

cm ^repo ^create Rep01

cm ^repo ^mk list

== CMD\_DESCRIPTION\_REPOSITORY\_DELETE ==

Deletes a repository from a server.

== CMD\_USAGE\_REPOSITORY\_DELETE ==

Usage:

cm ^repository | ^repo ^delete | ^rm <repspec>

Options:

repspec Repository specification.

(Use 'cm ^help ^objectspec' to learn more about rep specs.)

== CMD\_HELP\_REPOSITORY\_DELETE ==

Remarks:

Deletes a repository from the repository server.

The data is not removed from the database backend, but unplugged

so that it will not be accessible anymore.

(Data can be reconnected afterwards, see 'cm ^repository ^add'.)

Examples:

cm ^repository ^delete myrepository@^repserver:myserver:8084

cm ^repository ^rm myrepository@myserver:8084

cm ^repo ^rm myrepository

== CMD\_DESCRIPTION\_REPOSITORY\_LIST ==

Lists the repositories on a server.

== CMD\_USAGE\_REPOSITORY\_LIST ==

Usage:

cm ^repository | ^repo [^list | ^ls] [<repserverspec>] [--^format=<str\_format>]

Options:

repserverspec Repository server specification.

(Use 'cm ^help ^objectspec' to learn more about rep server

specs.)

--^format Retrieves the output message in a specific format. See

Remarks for more info.

== CMD\_HELP\_REPOSITORY\_LIST ==

Remarks:

This command accepts a format string to show the output.

The output parameters of this command are the following:

{^repid} | {0} Repository identifier.

{^repname} | {1} Repository name.

{^repserver} | {2} Server name.

{^repowner} | {3} Repository owner.

{^repguid} | {4} Unique identifier of the repository.

{^tab} Inserts a tab space.

{^newline} Inserts a new line.

If the format parameter value is '^TABLE', the output will be printed

using a table format with the {^repid}, {^repname} and {^repserver} fields.

Examples:

cm ^repository

(Lists all repositories.)

cm ^repository ^list localhost:8084 --^format="{1, -20} {3}"

(Writes the repository name in 20 spaces, aligned to left, then a blank,

and then the repository owner.)

cm ^repository ^ls localhost:8084 --^format="{^repname, -20} {^repowner}"

(Writes the same as the previous example.)

cm ^repo ^ls localhost:8084 --^format=^TABLE

(Writes the list of repositories using a table format with the following

fields: repository id, repository name, and repository server name.)

== CMD\_DESCRIPTION\_REPOSITORY\_RENAME ==

Renames a repository.

== CMD\_USAGE\_REPOSITORY\_RENAME ==

Usage:

cm ^repository | ^repo ^rename [<repspec>] <new\_name>

repspec Repository to be renamed.

(Use 'cm ^help ^objectspec' to learn more about repository

specifications.)

new\_name New name for the repository.

== CMD\_HELP\_REPOSITORY\_RENAME ==

Remarks:

This command renames a repository.

If no repspec is specified, current repository will be assumed.

Examples:

cm ^repository ^rename development

(The current repository will be renamed to 'development'.)

cm ^repo ^rename ^rep:default@SERVER:8084 development

(The 'default' repository on 'SERVER' will be renamed to 'development'.)

== CMD\_DESCRIPTION\_REPOSITORY\_ADD ==

Connects an existing repository by adding its database.

== CMD\_USAGE\_REPOSITORY\_ADD ==

Usage:

cm ^repository | ^repo ^add <db\_file> <rep\_name> <repserverspec>

db\_file The name of the database file on the database backend.

rep\_name The name of the repository.

repserverspec The repository server specification.

(Use 'cm ^help ^objectspec' to learn more about repository

server specifications.)

== CMD\_HELP\_REPOSITORY\_ADD ==

Remarks:

Reconnects an existing repository database to the server.

Example: After using the 'cm ^repository ^delete' command, use the '^add'

command to move a repository from one server to another or to restore an

archived repository.

Examples:

cm ^repository ^add rep\_27 myrepository myserver:8084

== CMD\_DESCRIPTION\_TRIGGER\_CREATE ==

Creates a new trigger on a server.

== CMD\_USAGE\_TRIGGER\_CREATE ==

Usage:

cm ^trigger | ^tr ^create | ^mk <subtype-type> <new\_name> <script\_path>

[--^position=<new\_position>]

[--^filter=<str\_filter>]

[--^server=<repserverspec>]

subtype-type Trigger execution and trigger operation.

Type 'cm ^showtriggertypes' to see a list of trigger

types.

new\_name Name of the new trigger.

script\_path Disk path on the server where the script to execute is

located. If the command line starts with "^webtrigger ",

the trigger will be considered as a web trigger. See

Remarks for more information.

Options:

--^position New position of the specified trigger.

This position must not be in use by another trigger of

the same type.

--^filter Checks only items that matches the specified filter.

--^server Creates the trigger on the specified server.

If no server is specified, executes the command on the

one configured on the client.

(Use 'cm ^help ^objectspec' to learn more about repository

server specifications.)

== CMD\_HELP\_TRIGGER\_CREATE ==

Remarks:

Web triggers: A web trigger is created by typing "^webtrigger <target-uri>"

as the trigger command. In this case, the trigger will execute a POST query

against the specified URI -where the request body contains a JSON

dictionary with the trigger environment variables- and a fixed INPUT key

pointing to an array of strings.

Examples:

cm ^trigger ^create ^after-setselector "BackupMgr" "/path/to/script" --^position=4

cm ^tr ^mk ^before-mklabel new "/path/to/script" --^server=myserver:8084

cm ^tr ^mk ^after-mklabel Log "/path/to/script" --^filter="^rep:myRep,LB\*"

(This trigger will be executed only if the label name starts with 'LB'

and it is being created in a repository called 'myRep'.)

cm ^tr ^mk ^after-checkin NotifyTeam "^webtrigger http://myserver.org/api"

== CMD\_DESCRIPTION\_MOVE ==

Moves or renames a file or directory.

== CMD\_USAGE\_MOVE ==

Usage:

cm ^move | ^mv <src\_path> <dst\_path> [--^format=<str\_format>]

[--^errorformat=<str\_format>]

src\_path Source item path.

dst\_path Destination item path.

Options:

--^format Retrieves the output message in a specific format. See

Remarks for more info.

--^errorformat Retrieves the error message in a specific format. See

Remarks for more info.

== CMD\_HELP\_MOVE ==

Remarks:

This command moves or renames an item in the repository.

Changes are done in the local filesystem too.

If the source path is a file, the destination path can be a file or a

directory. In the first case, the file is renamed; otherwise, the item

is moved.

If source path is a directory, the destination path must be a directory.

The item to move or rename must exist.

Format:

{0} Source path (both for '--^format' and '--^errorformat')

{1} Destination path (both for '--^format' and '--^errorformat')

Examples:

cm ^move file.txt file.old

(Renames the item.)

cm ^mv .\file.old .\oldFiles

(Moves 'file.old' to 'oldFiles'.)

cm ^move .\src .\src2

(Renames a directory.)

== CMD\_DESCRIPTION\_LABEL ==

Allows the user to manage labels.

== CMD\_USAGE\_LABEL ==

Usage:

cm ^label | ^lb <command> [options]

Commands:

^create | ^mk

^delete | ^rm

^rename

To get more information about each command run:

cm ^label <command> --^usage

cm ^label <command> --^help

== CMD\_HELP\_LABEL ==

Examples:

cm ^label myNewLabel ^cs:42

('^create' command is optional.)

cm ^label ^rename myNewLabel newLabelName

cm ^label ^delete newLabelName

== CMD\_DESCRIPTION\_LABEL\_CREATE ==

Applies a label to a changeset and creates the label if required.

== CMD\_USAGE\_LABEL\_CREATE ==

Usage:

cm ^label [^create] <lbspec> [<csetspec> | <wk\_path>]

[--^allxlinkedrepositories]

[-^c=<str\_comment> | -^commentsfile=<comments\_file>]

lbspec The new label name.

(Use 'cm ^help ^objectspec' to learn more about label

specs.)

csetspec Name or full specification of the changeset to label.

(Use 'cm ^help ^objectspec' to learn more about changeset

specs.)

wk\_path Path of the workspace to label. (The changeset that the

workspace is pointing will be labeled.)

Options:

--^allxlinkedrepositories Creates the new label in all Xlinked repositories.

-^c Applies the specified comment to the new label.

-^commentsfile Applies the comment in the specified file to the

new label.

== CMD\_HELP\_LABEL\_CREATE ==

Remarks:

Set PLASTICEDITOR environment variable to specify an editor to type the

comment.

Examples:

cm ^label ^create ^lb:BL001 ^cs:1203 -^commentsfile=commentlb001.txt

(Creates label 'BL001' attached to changeset 1203, and applies the comment

in the 'commentlb001.txt' file.)

cm ^label BL002 ^cs:1203 -^c="first release"

(Creates label 'BL002', with a comment, and attached to changeset 1203.)

== CMD\_DESCRIPTION\_LABEL\_DELETE ==

Deletes one or more labels.

== CMD\_USAGE\_LABEL\_DELETE ==

Usage:

cm ^label ^delete <lbspec>[ ...]

lbspec Label to delete. Use a whitespace to separate labels.

(Use 'cm ^help ^objectspec' to learn more about label

specs.)

== CMD\_HELP\_LABEL\_DELETE ==

Remarks:

This command deletes one or more labels.

Examples:

cm ^label ^delete ^lb:BL001

(Deletes the label 'BL001'.)

cm ^label ^delete ^lb:BL001 ^lb:BL002@reptest@server2:8084

(Deletes the labels 'BL001' and 'BL002'.)

== CMD\_DESCRIPTION\_LABEL\_RENAME ==

Renames a label.

== CMD\_USAGE\_LABEL\_RENAME ==

Usage:

cm ^label ^rename <lbspec> <new\_name>

lbspec Label to rename.

(Use 'cm ^help ^objectspec' to learn more about label specs.)

new\_name New name for the label.

== CMD\_HELP\_LABEL\_RENAME ==

Remarks:

This command renames a label.

Examples:

cm ^label ^rename ^lb:BL001 BL002

(Renames the label 'BL001' to 'BL002'.)

== CMD\_DESCRIPTION\_OBLITERATE ==

DEPRECATED

== CMD\_USAGE\_OBLITERATE ==

DEPRECATED.

== CMD\_HELP\_OBLITERATE ==

DEPRECATED.

== CMD\_DESCRIPTION\_OBJECTSPEC ==

Describes how to write object specs.

== CMD\_USAGE\_OBJECTSPEC ==

Usage:

cm ^objectspec

To get all the information about how to build object specs.

== CMD\_HELP\_OBJECTSPEC ==

Several Plastic SCM commands expect 'object specs' as input to refer to a

given object (typically a branch, changeset, repository, etc).

This documentation describes the different "specs" available and how to

build them.

Each spec type begins with a unique tag, for example "^rep:" or "^cs:". The tag

must be specified for commands that take a general object spec, for example

"cm ^setowner object\_spec", but can often be omitted for commands that take only

a single type of spec, for example, "cm ^getfile revision\_spec".

-- Repository server spec (repserverspec) --

^repserver:name:port

Examples:

cm ^repo ^list ^repserver:skull:8084

cm ^repo ^list skull:8084

Side note:

We call it 'repository server spec', instead of just 'server spec' for

historical reasons. Long ago, we had separate workspace and repository

servers, and the naming survived.

-- Repository spec (repspec) --

^rep:rep\_name@[repserverspec]

Examples:

cm ^showowner ^rep:codice@localhost:6060

(Here the "^rep:" is required because ^showowner admits not only repos

but also other types of objects. So it needs the user to indicate the

object type.)

-- Branch spec (brspec) --

^br:[/]br\_name[@repspec]

Examples:

cm ^switch ^br:/main@^rep:plastic@^repserver:skull:9095

(In this case "^br:", "^rep" and "^repserver" are not needed, so the

command admits a much shorter form:

"cm ^switch main@plastic@skull:9095".)

cm ^find ^revisions "^where ^branch='^br:/main/task001'"

Remark:

The initial '/' on the branch is not mandatory. We used to specify all

our branches as /main, /main/task001, and so on. But now, we prefer the

shorter form main, main/task001 which makes commands more compact.

-- Changeset spec (csetspec) --

^cs:cs\_number|cs\_guid[@repspec]

The number or GUID of the changeset can be specified.

Examples:

cm ^ls /code --^tree=ae1390ed-7ce9-4ec3-a155-e5a61de0dc77@code@skull:7070

-- Label spec (labelspec) --

^lb:lb\_name[@repspec]

Examples:

cm ^switch ^lb:RELEASE2.0

cm ^switch ^lb:RELEASE1.4@myrep@MYSERVER:8084

-- Revision spec --

There are different types of rev specs:

^rev:item\_path[#(brspec|csetspec|labelspec)]

^rev:^serverpath:item\_path#(brspec|cset\_spec|lb\_spec)

^rev:^revid:rev\_id[@rep\_spec]

^rev:^itemid:item\_id#(br\_spec|cset\_spec|lb\_spec)

Examples:

cm ^diff ^rev:readme.txt#^cs:19 ^rev:readme.txt#^cs:20

cm ^diff ^serverpath:/doc/readme.txt#^cs:19@myrepo \

^serverpath:/doc/readme.txt#^br:/main@myrepo@localhost:8084

cm ^cat ^revid:1230@^rep:myrep@^repserver:myserver:8084

-- Item spec --

^item:path

Rarely used.

Example:

cm ^find ^revision "^where ^item='^item:.'"

-- Attribute spec --

^att:att\_name[@repspec]

Example:

cm ^attribute ^set ^att:merged@code@doe:8084 ^cs:25@code@doe:8084 done

-- Shelve spec --

^sh:sh\_number[@repspec]

Example:

cm ^diff ^sh:2 ^sh:4

-- Workspace specs --

^wk:name@clientmachine

Rarely used, since they only apply to workspace related commands. Useful to

specify the workspace by name and machine instead of path.

Examples:

cm ^showselector ^wk:codebase@modok

Side note:

These specs come from the old days of Plastic SCM 2.x where 'workspace

servers' existed as a way to store workspace metadata in a centralized

way. Were deprecated due to performance issues.

== CMD\_DESCRIPTION\_PARTIAL ==

Runs commands in a partial workspace.

== CMD\_USAGE\_PARTIAL ==

Usage:

cm ^partial <command> [options]

Commands:

^configure

^add

^undo

^co | ^checkout

^unco | ^undocheckout

^ci | ^checkin

^mv | ^move

^rm | ^remove

^stb | ^switch

^upd | ^update

To get more information about each command run:

cm ^partial <command> --^usage

cm ^partial <command> --^help

== CMD\_HELP\_PARTIAL ==

Examples:

cm ^partial ^configure +/background-blue.png

cm ^partial ^update landscape-1024.png

cm ^partial ^checkin eyes-green.png eyes-black.png

== CMD\_DESCRIPTION\_PARTIAL\_ADD ==

Adds an item to version control.

== CMD\_USAGE\_PARTIAL\_ADD ==

Usage:

cm ^partial ^add [-^R | -^r | --^recursive] [--^silent] [--^parents]

[--^ignorefailed] [--^skipcontentcheck] <item\_path>[ ...]

item\_path Items to add. Use double quotes (" ") to specify paths

containing spaces. Use a whitespace to separate paths.

Use \* to add all the contents of the current directory.

Options:

-^R Adds items recursively.

--^silent Does not show any output.

--^parents Includes the parent directories of the items specified

in the operation.

--^ignorefailed If an item cannot be added, the add operation will

continue without it. Note: If a directory cannot be

added, its content is not added.

--^skipcontentcheck When the extension is not enough to set the file as

text or binary, it will be set as binary by default

instead of checking the content to detect the type.

== CMD\_HELP\_PARTIAL\_ADD ==

Remarks:

Requirements to add items:

- The parent directory of the item to add must be previously added.

Examples:

cm ^partial ^add pic1.png pic2.png

(Adds 'pic1.png' and 'pic2.png' items.)

cm ^partial ^add c:\workspace\picture.png

(Adds 'picture.png' item in path 'c:\workspace'.)

cm ^partial ^add -^R c:\workspace\src

(Recursively adds 'src'.)

cm ^partial ^add --^parents samples\design01.png

(Adds 'design01.png' file and 'samples' parent folder.)

cm ^partial ^add -^R \*

(Recursively adds all the contents of the current directory.)

== CMD\_DESCRIPTION\_PARTIAL\_CHECKIN ==

Stores changes in the repository.

== CMD\_USAGE\_PARTIAL\_CHECKIN ==

Usage:

cm ^partial ^checkin | ^ci [<item\_path>[ ...]]

[-^c=<str\_comment> | -^commentsfile=<comments\_file>]

[--^all | -^a] [--^applychanged] [--^keeplock]

[--^symlink] [--^ignorefailed]

Options:

item\_path Items to checkin. Use double quotes (" ") to specify

paths containing spaces. Use a whitespace to separate

paths.

Use . to apply checkin to current directory.

-^c Specifies a comment to the changeset created in the

checkin operation.

-^commentsfile Applies the comment from the specified file to the

changeset created in the checkin operation.

--^all | -^a Includes also the items changed, moved, and deleted

locally on the specified paths.

--^applychanged Applies the checkin operation to the changed items

detected in the workspace along with the checked out

items.

--^keeplock Keeps the lock of the locked items after the checkin

operation.

--^symlink Applies the checkin operation to the symlink and not to

the target.

--^ignorefailed Any changes that cannot be applied (because the lock

- a.k.a. exclusive checkout - cannot be adquired or

because local changes are in conflict with the server

changes) are discarded and the checkin operation

continues without them.

== CMD\_HELP\_PARTIAL\_CHECKIN ==

Remarks:

- If <item\_path> is not specified, the checkin will involve all the

pending changes in the workspace.

- The checkin operation always applies recursively from the given path.

- To checkin an item:

- The item must be under source code control.

- The item must be checked out.

- If the item is changed but not checked out, the '--^applychanged' flag

is not necessary unless <item\_path> is a directory or it contains

wildcards ('\*').

Revision content should be different from previous revision in order to be

checked in.

Set PLASTICEDITOR environment variable to specify an editor to type the

comment.

Reading input from stdin:

The '^partial ^checkin' command can read paths from stdin. To do this, pass a

single dash "-".

Example: cm ^partial ^checkin -

Paths will be read until an empty line is entered.

This allows you to use pipe to specify which files to checkin.

Example:

dir /S /B \*.c | cm ^partial ^checkin --^all -

(In Windows, checkins all .c files in the workspace.)

Examples:

cm ^partial ^checkin figure.png landscape.png

(Applies the checkin to 'figure.png' and 'landscape.png' checked-out files.)

cm ^partial ^checkin . -^commentsfile=mycomment.txt

(Applies checkin to current directory and sets the comment from the

'mycomment.txt' file.)

cm ^partial ^ci background.png -^c="my comment" --^keeplock

(Applies the checkin to 'background.png', includes a comment, and keeps the

lock.)

cm ^partial ^checkin --^applychanged

(Applies the checkin to all pending changes in the workspace.)

cm ^partial ^checkin link --^symlink

(Applies the checkin to the link file and not to the target, available on

UNIX environments.)

cm ^partial ^checkin . --^ignorefailed

(Applies checkin to current directory, ignoring the changes that cannot be

applied.)

== CMD\_DESCRIPTION\_PARTIAL\_CHECKOUT ==

Marks files as ready to modify.

== CMD\_USAGE\_PARTIAL\_CHECKOUT ==

Usage:

cm ^partial ^checkout | ^co [<item\_path>[ ...]] [--^resultformat=<str\_format>]

[--^silent] [--^ignorefailed]

Options:

item\_path Items to checkout. Use double quotes (" ") to specify

paths containing spaces. Use a whitespace to separate

paths.

Use . to apply checkout to current directory.

--^resultformat Retrieves the output result message in a specific

format.

--^silent Does not show any output.

--^ignorefailed If an item cannot be locked (the exclusive checkout

cannot be performed), the checkout operation will

continue without it.

== CMD\_HELP\_PARTIAL\_CHECKOUT ==

Remarks:

To checkout an item:

- The item must be under source code control.

- The item must be checked in.

If locks are configured on the server (lock.conf exists), then each time

a checkout on a path happens, Plastic checks if it meets any of the rules

and if so, the path will be in exclusive checkout (locked) so that none can

simultaneously checkout.

You can get all the locks in the server by using 'cm ^lock ^list'.

Check the Administrator Guide to learn how locking works:

https://www.plasticscm.com/download/help/locking

Examples:

cm ^partial ^checkout pic1.png pic2.png

(Checkouts 'pic1.png' and 'pic2.png' files.)

cm ^partial ^co \*.png

(Checkouts all png files.)

cm ^partial ^checkout .

(Checkouts current directory.)

cm ^partial ^checkout -^R c:\workspace\src

(Recursively checkouts 'src' folder.)

== CMD\_DESCRIPTION\_PARTIAL\_CONFIGURE ==

Allows you to configure your workspace by loading or unloading items from it.

== CMD\_USAGE\_PARTIAL\_CONFIGURE ==

Usage:

cm ^partial ^configure <+|-path>[ ...] [--^silent] [--^ignorefailed]

[--^ignorecase] [--^restorefulldirs]

path Paths to be loaded or unloaded. Use double quotes (" ") to

specify paths containing spaces. Use a whitespace to separate

paths.

Paths have to start with "/".

Options:

--^silent Does not show any output.

--^ignorefailed Skips all errors during the process. Incorrect paths

will not cause the command to stop.

--^ignorecase Ignores casing on the paths. With this flag, '^configure'

will work for "/Data/Textures" even if the user writes

"/data/teXtures".

--^restorefulldirs Resets an invalid directory configuration (happens when

a non-partial operation is run on a partial workspace).

The directories in this list will be fully configured

(full check) which means they will automatically

download new content during the update.

This operation does not download any files, just

restores the directory configuration on partial

workspaces.

== CMD\_HELP\_PARTIAL\_CONFIGURE ==

Remarks:

The command assumes recursive operation.

Examples:

cm ^partial ^configure +/landscape\_grey.png

(Loads 'landscape\_grey.png' item.)

cm ^partial ^configure -/landscape\_black.png

(Unloads 'landscape\_black.png' item.)

cm ^partial ^configure +/soft -/soft/soft-black.png

(Loads all 'soft' directory children items except 'soft-black.png'.)

cm ^partial ^configure -/

(Unloads the whole workspace.)

cm ^partial ^configure -/ +/

(Loads the whole workspace.)

cm ^partial ^configure -/figure-64.png --^ignorefailed

(Unloads 'figure-64.png' item even if it was already unloaded.)

cm ^partial ^configure +/ --^restorefulldirs

(Sets all directories to automatically download the new content.)

cm ^partial ^configure +/src/lib --^restorefulldirs

(Sets only '/src/lib' and its subdirectories to automatically download the

new content.)

== CMD\_DESCRIPTION\_PARTIAL\_MOVE ==

Moves or renames a file or directory.

== CMD\_USAGE\_PARTIAL\_MOVE ==

Usage:

cm ^partial ^move | ^mv <src\_path> <dst\_path> [--^format=<str\_format>]

src\_path Source item path.

dst\_path Destination item path.

Options:

--^format Retrieves the output message in a specific format. See

Remarks for more info.

== CMD\_HELP\_PARTIAL\_MOVE ==

Remarks:

This command moves or renames an item in the repository.

Changes are done in the local filesystem too.

If the source path is a file, the destination path can be a file or a

directory. In the first case, the file will be renamed; otherwise, the item

will be moved.

If source path is a directory, the destination path must be a directory.

The item to move or rename must exist.

Format:

{0} Source path.

{1} Destination path.

Examples:

cm ^partial ^move file.png file-blue.png

(Renames the item.)

cm ^partial ^mv .\file-blue.png .\blueFiles

(Moves 'file-blue.png' to 'blueFiles'.)

cm ^partial ^move .\design .\marketing

(Renames a directory.)

== CMD\_DESCRIPTION\_PARTIAL\_RM ==

Deletes a file or directory from version control.

== CMD\_USAGE\_PARTIAL\_RM ==

Usage:

cm ^partial ^remove | ^rm <item\_path>[ ...] [--^nodisk]

item\_path Items path to remove. Use double quotes (" ") to

specify paths containing spaces. Use a whitespace to separate

paths.

Options:

--^nodisk Removes from version control, but keeps the item on disk.

== CMD\_HELP\_PARTIAL\_RM ==

Remarks:

Items are deleted from disk. Removed items are removed from the parent

directory in the source code control.

Requirements:

- The item must be under source code control.

Examples:

cm ^partial ^remove src

(Removes 'src'. If 'src' is a directory, this is the same that:

cm ^partial ^remove -^R src.)

cm ^partial ^remove c:\workspace\pic01.png --^nodisk

(Removes 'pic01.png' from version control, but keeps it on disk.)

== CMD\_DESCRIPTION\_PARTIAL\_SWITCH ==

Sets a branch as the working branch.

== CMD\_USAGE\_PARTIAL\_SWITCH ==

Usage:

cm ^switch <branch\_spec> [--^report | --^silent] [--^workspace=<path>]

(Sets the working branch and updates the workspace.)

cm ^switch <branch\_spec> --^configure <+|-path>[ ...] [--^silent]

[--^ignorefailed] [--^ignorecase] [--^workspace=<path>]

(Sets the working branch and runs a workspace configuration like the 'cm

^partial ^configure' command does.)

branch\_spec Branch specification. (Use 'cm ^help ^objectspec' to learn

more about branch specs.)

path Paths to be loaded or unloaded. Use double quotes (" ")

to specify paths containing spaces. Use a whitespace to

separate paths. Paths must start with "/".

Options:

--^silent Does not show any output.

--^report Prints a list of the applied changes when the command

is finished. Using '--^silent' will override this setting.

This option only works when the '--^configure' option

is not specified.

--^configure Configures (loads / unloads items) the workspace

after updating the working branch. Check 'cm ^partial

^configure --^help' to learn how to specify the paths

to configure.

--^ignorefailed Skips all errors during the configuration process.

Incorrect paths will not cause the command to stop.

--^ignorecase Ignores casing on the paths. With this flag, option

'--^configure' works for "/Data/Textures" even if the user

writes "/data/teXtures".

--^workspace=path Path where the workspace is located.

== CMD\_HELP\_PARTIAL\_SWITCH ==

Remarks:

This command allows users to update the working branch. After updating the

branch, the command updates the workspace to the new branch as the

'cm ^partial ^update' command would do. However, if the '--^configure' option is

specified, the command allows to configure the workspace using the new

branch configuration as the 'cm ^partial ^configure' command would do.

Examples:

cm ^switch ^br:/main/task

(Sets /main/task as working branch and updates the workspace.)

cm ^switch ^br:/main/task --^configure +/art/images

(Sets /main/task as working branch and configures the workspace to

load the /art/images folder.)

== CMD\_DESCRIPTION\_PARTIAL\_UNCO ==

Undoes the checkout on an item.

== CMD\_USAGE\_PARTIAL\_UNCO ==

Usage:

cm ^partial ^undocheckout | ^unco <item\_path>[ ...] [--^silent]

item\_path Items to apply the operation. Use double quotes (" ")

to specify paths containing spaces. Use a whitespace to

separate paths.

Use . to apply the operation to current directory.

Options:

--^silent Does not show any output.

== CMD\_HELP\_PARTIAL\_UNCO ==

Remarks:

If an item is checked-out and you do not want to checkin it, you can undo

the checkout using this command. Both files and folders can be unchecked

out. The item will be updated to the state it had before checking it out.

Requirements:

- The item must be under source code control.

- The item must be checked out.

Examples:

cm ^partial ^undocheckout .

(Undoes checkouts in the current directory.)

cm ^partial ^undocheckout pic1.png pic2.png

cm ^unco c:\workspace\design01.png

(Undoes checkouts of the selected files.)

== CMD\_DESCRIPTION\_PARTIAL\_UNDO ==

Undoes changes in a workspace.

== CMD\_USAGE\_PARTIAL\_UNDO ==

Usage:

cm ^partial ^undo [<path>[ ...]] [--^symlink] [-^r | --^recursive]

[<filter>[ ...]]

[--^silent | --^machinereadable [--^startlineseparator=<sep>]

[--^endlineseparator=<sep>] [--^fieldseparator=<sep>]]

path Path of the files or directories to apply the

operation to. Use double quotes (" ") to specify

paths containing spaces. Use a whitespace to

separate paths.

If no path is specified, by default the undo

operation will take all of the files in the current

directory.

filter Applies the specified filter or filters to the given

paths. Use a whitespace to separate filters. See the

Filters section for more information.

Options:

--^symlink Applies the undo operation to the symlink and not

to the target.

-^r Executes the undo recursively.

--^silent Does not show any output.

--^machinereadable Outputs the result in an easy-to-parse format.

--^startlineseparator Used with the '--^machinereadable' flag, specifies

how the lines should start.

--^endlineseparator Used with the '--^machinereadable' flag, specifies

how the lines should end.

--^fieldseparator Used with the '--^machinereadable' flag, specifies

how the fields should be separated.

Filters:

If no flag is specified, by default, all changes are undone, but the

paths can be filtered using one or more of the flags below.

If a file or directory matches one or more of the specified kinds of change,

all of the changes on said file or directory will be undone.

For example, if you specify both '--^checkedout' and '--^moved', if a file is

both checkedout and moved, both changes will be undone.

--^checkedout Select checked-out files and directories.

--^unchanged Select files whose content is unchanged.

--^changed Select locally changed or checked-out files and

directories.

--^deleted Select deleted files and directories.

--^moved Select moved files and directories.

--^added Select added files and directories.

== CMD\_HELP\_PARTIAL\_UNDO ==

Remarks:

The ^undo command is dangerous - it undoes work in an irreversible way.

Once the ^undo has finished, it is not possible to recover the previous state

of the files and directories affected by it. If no path is specified

in the arguments, by default it will undo every change in the current

directory, but not recursively.

These are equivalent when executed from the /src directory:

/src

|- file.txt

|- code.cs

\- /test

|- test\_a.py

\- test\_b.py

cm ^partial ^undo

cm ^partial ^undo \*

cm ^partial ^undo file.txt code.cs /test

cm ^partial ^undo .

cm ^partial ^undo /src file.txt code.cs

If you want the operation to be recursive, you must specify the '-^r' flag.

To undo all of the changes below a directory (including changes affecting

the directory itself):

cm ^partial ^undo dirpath -^r

If dirpath is a workspace path, every change in the workspace will be

undone.

Examples:

cm ^partial ^undo . -^r

(Undoes all changes in the current directory recursively. If executed

from the workspace's root, undoes all changes in the entire workspace.)

cm ^partial ^co file.txt

cm ^partial ^undo file.txt

(Undoes the checkout on file.txt.)

^echo ^content >> file.txt

cm ^partial ^undo file.txt

(Undoes the local change to file.txt.)

cm ^partial ^undo src

(Undoes changes to the src directory and its files.)

cm ^partial ^undo src/\*

(Undo changes in every file and directory contained in src, without

affecting src.)

cm ^partial ^undo \*.cs

(Undo changes to every file or directory that matches \*.cs in the current

directory.)

cm ^partial ^undo \*.cs -^r

(Undoes changes on every file or directory that matches \*.cs in the current

directory and every directory below it.)

cm ^partial ^co file1.txt file2.txt

^echo ^content >> file1.txt

cm ^partial ^undo --^unchanged

(Undoes the checkout of unchanged file2.txt, ignoring locally changed

file1.txt.)

^echo ^content >> file1.txt

^echo ^content >> file2.txt

cm ^partial ^co file1.txt

cm ^partial ^undo --^checkedout

(Undoes the changes in checked-out file file1.txt, ignoring file2.txt as it is

not checked-out.)

cm ^partial ^add file.txt

cm ^partial ^undo file.txt

(Undoes the add of file.txt, making it once again a private file.)

^rm file1.txt

^echo ^content >> file2.txt

cm ^partial ^add file3.txt

cm ^partial ^undo --^deleted --^added \*

(Undoes the file1.txt delete and file3.txt add, ignoring the file2.txt

change.)

== CMD\_DESCRIPTION\_PARTIAL\_UPDATE ==

Updates the partial workspace and downloads latest changes.

== CMD\_USAGE\_PARTIAL\_UPDATE ==

Usage:

cm ^partial ^update [<item\_path>[ ...]] [--^changeset=<number>]

[--^silent | --^report] [--^dontmerge]

item\_path Items to be updated. Use double quotes (" ") to specify

paths containing spaces. Use a whitespace to separate

paths.

Use . to apply update to current directory.

If no path is specified, then the current partial

workspace is fully updated.

Options:

--^changeset Updates the partial workspace to a specific changeset.

--^silent Does not show any output.

--^report Prints a list of the applied changes when the command

is finished. Using '--^silent' will override this setting.

--^dontmerge Does not merge the file conflicts, it just skips them.

The other changes are properly applied. This option can

be useful for automation to avoid user interaction.

== CMD\_HELP\_PARTIAL\_UPDATE ==

Remarks:

The '^partial ^update' command updates the out-of-date files.

The command assumes recursive operation.

If all the specified paths are files inside the same Xlink when using the

'--^changeset' option, then the versions to download are searched in the

specified changeset of the Xlinked repository.

Examples:

cm ^partial ^update

(Updates all in the current partial workspace.)

cm ^partial ^update .

(Updates all current directory children items.)

cm ^partial ^update backgroud-blue.png

(Updates 'backgroud-blue.png' item.)

cm ^partial ^update soft\_black.png soft-grey.png

(Updates 'soft\_black.png' and 'soft-grey.png' items.)

cm ^partial ^update src --^report

(Updates all 'src' directory children items, printing the applied changes

list at the end.)

cm ^partial ^update src --^changeset=4

(Updates all 'src' directory children items to the content they loaded

in the changeset 4.)

cm ^partial ^update xlink/first.png --^changeset=4

(Updates 'xlink/first.png' item to the content it loaded in the changeset 4

of the Xlinked repository.)

== CMD\_DESCRIPTION\_PATCH ==

Generates a patch file from a spec or applies a generated patch to the current

workspace.

== CMD\_USAGE\_PATCH ==

Usage:

cm ^patch <source\_spec> [<source\_spec>] [--^output=<output\_file>]

[--^tool=<path\_to\_diff>]

Generates a patch file that contains the differences of a branch,

a changeset, or the differences between changesets. It also tracks

differences of text and binary files.

cm ^patch --^apply <patch\_file> [--^tool=<path\_to\_patch>]

Allows to apply the contents of a generated patch file in the current

workspace.

source\_spec Full spec of a changeset or a branch. (Use

'cm ^help ^objectspec' to learn more about specs.)

output\_file File to save the patch content. It no file is specified,

the patch content will be printed on standard output.

patch\_file Patch file to apply in the current workspace.

Options:

--^output Sets the output file of the patch command.

--^tool Sets the application to use (diff or patch).

== CMD\_HELP\_PATCH ==

Limitations:

If the output patch file already exists, the command will not overwrite it.

When applying a patch, the command will not apply changes to modified files

if they are not present on disk.

Important:

This command requires Diff and Patch tools, publicly available at

http://gnuwin32.sourceforge.net/packages/patch.htm and

http://gnuwin32.sourceforge.net/packages/diffutils.htm

Once installed, it's recommended to add their location to the PATH

environment variable.

Examples:

cm ^patch ^cs:4@default@localhost:8084

(Prints on console the differences of cset 4 in unified format.)

cm ^patch ^br:/main --^output=file.patch

(Generates a patch file with the differences of branch "main".)

cm ^patch ^br:/main --^output=file.patch --^tool=C:\gnu\diff.exe

(Same as above, using a custom exe.)

cm ^patch ^cs:2@default ^cs:4@default

(Prints on console the differences between csets 2 and 4 in unified format.)

cm ^patch --^apply file.patch --^tool=C:\gnu\patch.exe

(Applies the patch in 'file.patch' to the local workspace with a custom exe.)

== CMD\_DESCRIPTION\_QUERY ==

Executes SQL queries. Requires SQL storage.

== CMD\_USAGE\_QUERY ==

Usage:

cm ^query <sql\_command> [--^outputfile=<output\_file>]

[--^solveuser=<column\_name>[,...]]

[--^solvepath=<column\_name>[,...]]

[--^columnwidth=<value>] [--^nocolumnname]

[--^columnseparator=<sep>] [--^repository=<name>]

sql\_command The sql query to be executed.

Options:

--^outputfile Writes the result in an output file.

--^solveuser Sets the specified columns as username columns. The

query interpreter will assume that data of these columns

will be users, and will try to solve them.

--^solvepath Sets the specified columns as itemid column. The query

interpreter will try to solve item id to filesystem

paths.

--^columnwidth Specifies the width of each column to format the output.

--^nocolumnname Does not print column name.

--^columnseparator Uses char as column separator instead of a tab (\t).

--^repository Repository to query.

== CMD\_HELP\_QUERY ==

Remarks:

This command allows users to execute SQL queries in the server database.

In order to write SQL queries, use these two pre-defined functions to manage

users and paths:

- '^SolveUser(<username>)' that resolves a username into Plastic SCM format.

- '^SolvePath(<path>)' that resolves a disk path into an item id.

Also, you can use options to show query results in a human readable form.

You can use the options '--^solveuser=<column\_name>' and

'--^solvepath=<column\_name>' to specify columns that query interpreter

must convert to a legible text. You can specify more than one column name,

comma separated.

Examples:

cm ^query "^SELECT \* ^FROM ^revision" --^columnwidth=25 --^repository=reptest

(Retrieves data from 'revision' table from repository 'reptest'.)

cm ^query "^SELECT b.^sname ^as br\_name, o.^dtimestamp ^as date ^from ^branch b, \

^object o, ^seid s ^where b.^iobjid=o.^iobjid ^and o.^fidowner=s.^iseidid ^and \

s.^scode='^SolveUser(john)'" --^outputfile=query.txt

(Outputs into a file the branches with owner 'john'.)

cm ^query "^select r.^iobjid, r.^fiditem ^as path, s.^scode ^as username ^FROM \

^revision r, ^object o, ^seid s ^WHERE r.^iobjid=o.^iobjid ^and \

o.^fidowner=s.^iseidid ^and o.^dtimestamp>04/25/2014" \

--^solveuser=username --^solvepath=path --^repository=reptest@server2:9095

(Retrieves selected data from selected repository.)

cm ^query "^SELECT \* ^FROM ^revision ^WHERE ^fiditem=^SolvePath(c:\mywkpath\info)"

(Retrieves all revision data of path 'info'.)

== CMD\_DESCRIPTION\_ATTRIBUTE\_DELETE ==

Deletes one or more attributes.

== CMD\_USAGE\_ATTRIBUTE\_DELETE ==

Usage:

cm ^attribute | ^att ^delete | ^rm <att\_spec>[ ...]

att\_spec Attributes to delete. Use a whitespace to separate

attributes.

(Use 'cm ^help ^objectspec' to learn more about attribute

specs.)

== CMD\_HELP\_ATTRIBUTE\_DELETE ==

Remarks:

This command removes one or more attributes.

Examples:

cm ^attribute ^delete ^att:status

(Deletes the attribute 'status'.)

cm ^att ^rm status ^att:integrated@reptest@server2:8084

(Deletes the attributes 'status' and 'integrated'.)

== CMD\_DESCRIPTION\_ATTRIBUTE\_UNSET ==

Unsets an object's attribute.

== CMD\_USAGE\_ATTRIBUTE\_UNSET ==

Usage:

cm ^attribute | ^att ^unset <att\_spec> <object\_spec>

att\_spec Attribute specification. (Use 'cm ^help ^objectspec' to

learn more about attribute specs.)

object\_spec Specification of the object to remove the attribute

from. Attributes can be set on: branches, changesets,

shelvesets, labels, items, and revisions.

(Use 'cm ^help ^objectspec' to learn more about specs.)

== CMD\_HELP\_ATTRIBUTE\_UNSET ==

Remarks:

The command unsets an attribute that was previously set on an object. It

does not delete the attribute object itself.

Examples:

cm ^attribute ^unset ^att:status ^br:/main/SCM105

(Removes attribute realization 'status' from branch 'main/SCM105'.)

cm ^att ^unset ^att:integrated@reptest@localhost:8084 ^cs:25@reptest@localhost:8084

(Removes attribute realization 'integrated' from changeset 25, all in

repository 'reptest'.)

== CMD\_DESCRIPTION\_ATTRIBUTE\_RENAME ==

Renames an attribute.

== CMD\_USAGE\_ATTRIBUTE\_RENAME ==

Usage:

cm ^attribute | ^att ^rename <att\_spec> <new\_name>

att\_spec Attribute to rename. (Use 'cm ^help ^objectspec' to learn

more about attribute specs.)

new\_name New name for the attribute.

== CMD\_HELP\_ATTRIBUTE\_RENAME ==

Remarks:

This command renames an attribute.

Examples:

cm ^attribute ^rename ^att:status state

(Renames the attribute 'status' to 'state'.)

== CMD\_DESCRIPTION\_ATTRIBUTE\_EDIT ==

Edits the comment of an attribute.

== CMD\_USAGE\_ATTRIBUTE\_EDIT ==

Usage:

cm ^attribute | ^att ^edit <att\_spec> <new\_comment>

att\_spec Attribute to change its comment. (Use 'cm ^help ^objectspec'

to learn more about attribute specs.)

new\_comment New comment for the attribute. You can also specify a

default list of values for the attribute.

== CMD\_HELP\_ATTRIBUTE\_EDIT ==

Remarks:

This command changes the comment of an attribute.

To specify a default list of values for the attribute, you just need to

include a line like the following in the attribute comment:

'default: value\_one, "value two", value3, "Final value"'.

Examples:

cm ^attribute ^edit ^att:status "The status of a branch in the CI pipeline."

(Edits the comment of the attribute 'status'.)

cm ^attribute ^edit ^att:status "Status of a branch. default: open, resolved, reviewed"

(Edits the comment of the attribute 'status'. And also specifies a list of

values. So when you set the attribute 'status' to an object, you can select

one of the following values: "open", "resolved", or "reviewed".)

== CMD\_DESCRIPTION\_REPLICATE ==

WARNING: This command is deprecated.

Use 'cm ^pull' (equivalent to '^replicate') and 'cm ^push' (equivalent to

'^replicate --^push').

== CMD\_USAGE\_REPLICATE ==

== CMD\_HELP\_REPLICATE ==

== CMD\_DESCRIPTION\_PULL ==

Pulls a branch from another repo.

== CMD\_USAGE\_PULL ==

Usage:

cm ^pull <src\_br\_spec> <dst\_rep\_spec>

[--^preview] [--^nodata] [TranslateOptions]

[--^user=<usr\_name> [--^password=<pwd>] | AuthOptions]

(Direct server-to-server replication. Pulls a branch from a repository.)

cm ^pull <dst\_rep\_spec> --^package=<pack\_file> [AuthOptions]

(Package based replication. Imports the package in the destination repository.)

cm ^pull ^hydrate <dst\_br\_spec> [<src\_rep\_spec>]

[--^user=<usr\_name> [--^password=<pwd>] | AuthOptions]

(Introduces the missing data for all the changesets of a branch previously

replicated with '--^nodata'. If a repo to obtain the data is not specified,

Plastic tries to use the "replication source" (origin of the replicated

branch)).

cm ^pull ^hydrate <dst\_cs\_spec> [<src\_rep\_spec>]

[--^user=<usr\_name> [--^password=<pwd>] | AuthOptions]

(Introduces the missing data for a changeset previously replicated with

'--^nodata'. If a repo to obtain the data is not specified, Plastic tries to

use the "replication source").

src\_br\_spec The branch to pull from a remote repository.

(Use 'cm ^help ^objectspec' to learn more about branch specs.)

dst\_br\_spec The branch to hydrate.

(Use 'cm ^help ^objectspec' to learn more about branch specs.)

dst\_cs\_spec The changeset to hydrate.

(Use 'cm ^help ^objectspec' to learn more about changeset

specs.)

dst\_rep\_spec The destination repository.

(Use 'cm ^help ^objectspec' to learn more about repository

specs.)

--^package Specifies the previously created package file to import

for package based replication.

Useful to move data between servers when there is no

direct network connection.

Refer to 'cm ^push' to create a package file.

Options:

--^preview Gives information about what changes will be pulled but

no changes are actually performed. This option is useful

to check the data that will be transferred before

replicating changes.

--^nodata Replicates the branch changes without replicating the

data. This option is not allowed with package

replication.

TranslateOptions See the Translate options section for more information.

--^user, --^password Credentials to use if the authentication mode is

different in source and destination and there is not a

profile to authenticate to destination.

AuthOptions See the Authentication options section for more

information.

Translate options:

--^trmode=(^copy|^name|^table --^trtable=<translation\_table\_file>)

The source and destination repositories may use different authentication

modes. The '--^trmode' option specifies how to translate the user names from

the source to the destination. The '--^trmode' must be one of the following

values:

^copy (Default). Means that the user identifiers will be just copied.

^name The user identifiers will be matched by name.

^table Uses a translation table specified in the option '--^trtable'

(see below).

--^trtable=<translation\_table\_file>

If the translation mode is 'table', a translation table is a file

containing lines in the form <oldname;newname> (one per line). When the

branch is written to the destination repository, the objects created by

a user identified by "oldname" in the source repository will be set

to the user with "newname" on the destination.

Authentication options:

Authentication data can be specified using one of the two following modes:

1) Using authentication parameters: --^authmode=<mode> --^authdata=<data>

--^authmode=(^NameWorkingMode|^LDAPWorkingMode|^ADWorkingMode|^UPWorkingMode)

Examples:

(^LDAPWorkingMode) --^authdata=::0:dave:fPBea2rPsQaagEW3pKNveA

(^UPWorkingMode) --^authdata=dave:fPBea2rPsQaagEW3pKNveA==

The '--^authdata' line is the content of the <^SecurityConfig> entry

in the client.conf file and the profiles.conf file. The profiles.conf

file can be easily generated from the Plastic GUI in the replication

profiles tab under Preferences.

If you are using ^UPWorkingMode, you can simply specify:

--^authmode=^UPWorkingMode --^user=<user> --^password=<psw>

2) Authentication file where you may have a different file for each server

you connect to, containing the credentials for that server.

--^authfile=<authentication\_file>

The file contains 2 lines:

Line 1) mode, as described in '--^authmode'

Line 2) authentication data, as described in '--^authdata'

== CMD\_HELP\_PULL ==

Remarks:

The '^pull' command is able to replicate branches (along with their

changesets) between a source repository and a destination repository.

The repositories can be located at different servers.

There are two replication operations: '^push' and '^pull'.

A '^pull' operation means that the replication operation will demand data

from the source repository to be stored into the destination repository.

The client will connect to the destination repository and, from that host,

it will establish a connection to the source repository to retrieve the

targeted data. During pull it is the destination server which will be

connected to the source.

Although in a typical distributed scenario a developer pushes data from his

local server to the main server, the developer might want to pull the latest

repository updates from the main server, too.

Replication can resolve situations where concurrent changes have been made

on the same branch on two replicated repositories:

- Push: If you try to push your data to a repository having newer changes

than those you are sending, the system will ask you to pull the latest

changes, resolve the merge operation and, finally, try to push again.

- Pull: Whenever you pull changesets from a remote branch, they will be

correctly linked to their parent changesets. If the changeset you pulled

is not a child of the last changeset in the branch, then a multi-headed

scenario will appear. The branch will have more than one 'head', or last

changeset on the branch. You will need to merge the two 'heads' before

being able to push again.

Pull can work in two modes:

1) Direct communication between servers: The destination server will fetch

the data from the source server, automatically synchronizing data for

the specified branch.

2) Import a previously generated package with push and the '--^package' option.

Mode 1) requires the user running the command to be authenticated

by the remote server, either using the default authentication in the

client.conf file, or specifiying the '--^authmode' and '--^authdata' modifiers,

or '--^authmode' and '--^user' and '--^password' if the authentication mode is

^UPWorkingMode.

Mode 2) requires using a package file previously generated with the push

command.

Keep in mind that pull replication works in an indirect way. When executed,

the command asks the destination repository to connect to the source and

obtain the selected branch.

However, this can be done directly by using the push command.

This will make the command replicate the selected branch from source to

destination.

Examples:

cm ^pull ^br:/main@project1@remoteserver:8084 projectx@myserver:8084

(Pulls the 'main' branch from 'remoteserver' to 'myserver'. In this case,

both servers are configured with the same authentication mode.)

cm ^pull ^br:/main@project1@remoteserver:8084 projectx@myserver:8084 \

--^authmode=^LDAPWorkingMode --^authdata=::0:dave:fPBea2rPsQaagEW3pKNveA

(Pulls the same branch as before, but now the remote server is configured

to authenticate users with Active Directory. For instance, I am connecting

from a Linux machine to a Windows server configured to use Active Directory

integrated mode. I will specify my Active Directory user and cyphered

password and pass it as LDAP to the server.)

cm ^pull ^br:/main@project1@remoteserver:8084 projectx@myserver:8084 \

--^authmode=^UPWorkingMode --^user=dave --^password=mysecret

(Pulls the same branch, but now users are authenticated on the remote

server, taking advantage of the user/password database included in

Plastic SCM.)

cm ^pull ^br:/main@project1@remoteserver:8084 projectx@myserver:8084 --^nodata

(Replicates the 'main' branch from 'remoteserver' to 'myserver' without data.)

cm ^pull ^hydrate ^br:/main@projectx@myserver:8084 projectx@remoteserver:8084

(Hydrates all the changesets in the 'main' branch obtaining the data from

the remote server.)

cm ^pull ^hydrate ^cs:122169@projectx@myserver:8084 projectx@remoteserver:8084

(Hydrates changeset 122169 in 'myserver' obtaining the data from the remote

server.)

== CMD\_DESCRIPTION\_PUSH ==

Pushes a branch to another repo.

== CMD\_USAGE\_PUSH ==

Usage:

cm ^push <src\_br\_spec> <dst\_rep\_spec>

[--^preview] [TranslateOptions]

[--^user=<usr\_name> [--^password=<pwd>] | AuthOptions]

(Direct server-to-server replication. Pushes a branch from a repository.)

cm ^push <src\_br\_spec> --^package=<pack\_file> [AuthOptions]

(Package based replication. Creates a replication package in the source

server with the selected branch.)

src\_br\_spec The branch to push to a remote repository.

(Use 'cm ^help ^objectspec' to learn more about branch specs.)

dst\_rep\_spec The destination repository.

(Use 'cm ^help ^objectspec' to learn more about repository

specs.)

--^package Specifies path for exporting replication package for

package based replication.

Useful to move data between servers when there is no

direct network connection.

Options:

--^preview Gives information about what changes will be pushed,

but no changes are actually performed. This option is

useful to check the data that will be transferred before

replicating changes.

TranslateOptions See the Translate options section for more information.

--^user, --^password Credentials to use if the authentication mode is

different in source and destination and there is not a

profile to authenticate to destination.

AuthOptions See the Authentication options section for more

information.

Translate options:

--^trmode=(^copy|^name|^table --^trtable=<translation\_table\_file>)

The source and destination repositories may use different authentication

modes. The '--^trmode' option specifies how to translate the user names

from the source to the destination. The '--^trmode' must be one of the

following values:

^copy (Default). Means that the user identifiers will be just copied.

^name The user identifiers will be matched by name.

^table Uses a translation table specified in the option '--^trtable'

(see below).

--^trtable=<translation\_table\_file>

If the translation mode is 'table', a translation table is a file

containing lines in the form <oldname;newname> (one per line). When the

branch is written to the destination repository, the objects created by

a user identified by "oldname" in the source repository will be set

to the user with "newname" on the destination.

Authentication options:

Authentication data can be specified using one of the two following modes:

1) Using authentication parameters: --^authmode=<mode> --^authdata=<data>

--^authmode=(^NameWorkingMode|^LDAPWorkingMode|^ADWorkingMode|^UPWorkingMode)

Examples:

(^LDAPWorkingMode) --^authdata=::0:dave:fPBea2rPsQaagEW3pKNveA

(^UPWorkingMode) --^authdata=dave:fPBea2rPsQaagEW3pKNveA==

The '--^authdata' line is the content of the <^SecurityConfig> entry

in the client.conf file and the profiles.conf file. The profiles.conf

file can be easily generated from the Plastic GUI in the replication

profiles tab under Preferences.

If you are using ^UPWorkingMode, you can simply specify:

--^authmode=^UPWorkingMode --^user=<user> --^password=<psw>

2) Authentication file where you may have a different file for each server

you connect to, containing the credentials for that server.

--^authfile=<authentication\_file>

The file contains 2 lines:

Line 1) mode, as described in '--^authmode'

Line 2) authentication data, as described in '--^authdata'

== CMD\_HELP\_PUSH ==

Remarks:

The '^push' command is able to replicate branches (along with their

changesets) between a source repository and a destination repository.

The repositories can be located at different servers.

There are two replication operations: '^push' and '^pull'.

A '^push' operation means that the replication operation will send data

from the source repository to the destination repository. In this case,

the client will connect to the source repository, getting the data to

replicate, and then it will send it to the destination repository. While

the former (source) must have connectivity to the destination, the latter

(destination) will not connect itself to the source.

In a typical distributed scenario, a developer pushes data from his local

server to the main server. Also, the developer might want to pull the latest

repository updates from the main server, too.

Replication can resolve situations where concurrent changes have

been made on the same branch on two replicated repositories.

- Push: If you try to push your data to a repository having newer changes

than those you are sending, the system will ask you to pull the latest

changes, resolve the merge operation and, finally, try to push again.

- Pull: Whenever you pull changesets from a remote branch, they will be

correctly linked to their parent changesets. If the changeset you pulled

is not a child of the last changeset in the branch, then a multi-headed

scenario will appear. The branch will have more than one 'head', or last

changeset on the branch. You will need to merge the two 'heads' before

being able to push again.

Push can work in two modes:

1) Direct communication between servers: The origin server will send

the data to the destination server, automatically synchronizing data

for the specified branch.

2) Export package mode: The client will only connect to the source and

generate a replication package obtaining both data and metadata for the

specified branch. The '--^package' modifier will be used.

Both modes require the user running the command to be authenticated

by the server, either using the default authentication in the client.conf

file, or specifiying the '--^authmode' and '--^authdata' modifiers.

The ^push replication works in a direct way. When executed, the command

will replicate the selected branch from source to destination, instead of

asking the destination repository to connect to the source and obtain the

selected branch (as the pull does).

Examples:

cm ^push ^br:/main@project1@myserver:8084 projectx@remoteserver:8084

(Replicates the 'main' branch from 'myserver' to 'remoteserver'. In this case,

both servers are configured with the same authentication mode.)

cm ^push ^br:/main@project1@myserver:8084 projectx@remoteserver:8084 \

--^authmode=^LDAPWorkingMode --^authdata=::0:dave:fPBea2rPsQaagEW3pKNveA

(Replicates same branch as before, but now the remote server is configured

to authenticate users with Active Directory. For instance, I am connecting

from a Linux machine to a Windows server configured to use Active Directory

integrated mode. I will specify my Active Directory user and cyphered

password and pass it as LDAP to the server.)

cm ^push ^br:/main@project1@myserver:8084 projectx@remoteserver:8084 \

--^authmode=^UPWorkingMode --^user=dave --^password=mysecret

(Replicates the same branch, but now users are authenticated on the remote

server, taking advantage of the user/password database included in

Plastic SCM.)

== CMD\_DESCRIPTION\_CLONE ==

Clones a remote repository.

== CMD\_USAGE\_CLONE ==

Usage:

cm ^clone <src\_rep\_spec> [<dst\_rep\_spec> | <dst\_repserver\_spec>]

[--^user=<usr\_name> [--^password=<pwd>] | AuthOptions]

[TranslateOptions]

(Direct repository-to-repository clone.)

cm ^clone <src\_rep\_spec> --^package=<pack\_file>

[--^user=<usr\_name> [--^password=<pwd>] | AuthOptions]

(Clones to an intermediate package, that can be imported later using a

pull into the destination repository.)

src\_rep\_spec Source repository of the clone operation.

(Use 'cm ^help ^objectspec' to learn more about repository

specs.)

dst\_rep\_spec Destination repository of the clone operation. If it

exists, it must be empty. If it does not exist, it will

be created.

If it is not specified, the command will use user's

default repository server.

(Use 'cm ^help ^objectspec' to learn more about repository

specs.)

dst\_repserver\_spec Destination repository server of the clone operation.

If there is a repository with the same name as

<src\_rep\_spec> in the destination repository server, it

must be empty. If there is not, it will be created.

If it is not specified, the command will use user's

default repository server.

(Use 'cm ^help ^objectspec' to learn more about repository

server specs.)

Options:

--^user, --^password Credentials to use if the authentication mode is

different in source and destination and there is not a

profile to authenticate to destination.

--^package Exports the specified repository to a package file,

instead of a repository.

Useful for moving data between servers when there is no

direct network connection.

The resulting package must be imported using the

pull command.

TranslateOptions See the Translate options section for more information.

AuthOptions See the Authentication options section for more

information.

Translate options:

--^trmode=(^copy|^name|^table --^trtable=<translation\_table\_file>)

The source and destination repositories may use different authentication

modes. The '--^trmode' option specifies how to translate the user names from

the source to the destination. The '--^trmode' must be one of the following

values:

^copy (Default). Means that the user identifiers will be just copied.

^name The user identifiers will be matched by name.

^table Uses a translation table specified in the option '--^trtable'

(see below).

--^trtable=<translation\_table\_file>

If the translation mode is 'table', a translation table is a file

containing lines in the form <oldname;newname> (one per line). When the

branch is written to the destination repository, the objects created by

a user identified by "oldname" in the source repository will be set

to the user with "newname" on the destination.

Authentication options:

Authentication data can be specified using one of the two following modes:

1) Using authentication parameters: --^authmode=<mode> --^authdata=<data>

--^authmode=(^NameWorkingMode|^LDAPWorkingMode|^ADWorkingMode|^UPWorkingMode)

Examples:

(^LDAPWorkingMode) --^authdata=::0:dave:fPBea2rPsQaagEW3pKNveA

(^UPWorkingMode) --^authdata=dave:fPBea2rPsQaagEW3pKNveA==

The '--^authdata' line is the content of the <^SecurityConfig> entry

in the client.conf file and the profiles.conf file. The profiles.conf

file can be easily generated from the Plastic GUI in the replication

profiles tab under Preferences.

If you are using ^UPWorkingMode, you can simply specify:

--^authmode=^UPWorkingMode --^user=<user> --^password=<psw>

2) Authentication file where you may have a different file for each server

you connect to, containing the credentials for that server.

--^authfile=<authentication\_file>

The file contains 2 lines:

Line 1) mode, as described in '--^authmode'

Line 2) authentication data, as described in '--^authdata'

== CMD\_HELP\_CLONE ==

Remarks:

The clone command is able to replicate branches (along with their changesets,

labels, attributes, reviews, and so on) from a source repository to a

destination repository. The repositories can be located at different servers.

The destination repository can be created beforehand, but if it contains

previous data, the clone operation will fail.

The clone operation does NOT clone repository submodules, nor repositories

under a Xlink.

Examples:

cm ^clone awesomeProject@tardis@cloud

(Clones 'awesomeProject' repository from 'tardis@cloud' organization into

a local repository with the same name.)

cm ^clone repo@server.home:9095 repo-local

(Clones 'repo' from 'server.home:9095' into 'repo-local' at user's default

repository server.)

cm ^clone project@192.168.111.130:8084 ^repserver:192.168.111.200:9095

(Clones 'project' repository from '192.168.111.130:8084' into

'project@192.168.111.200:9095'.)

cm ^clone project@ldapserver:8084 --authfile=credentials.txt \

--^trmode=table --^trtable=table.txt

(Clones 'project' repository from 'ldapserver:8084' using an authentication

file against the remote repository, and translating users following the

specified translation table.)

cm ^clone project@server.home:9095 --^package=project.plasticpkg

cm ^repository ^create project@mordor.home:8084

cm ^pull --^package=project.plasticpkg project@mordor.home:8084

(Clones 'project' repository from 'server.home:9095' into the package

'project.plasticpkg', which is later imported through a pull into

the 'project' repository at 'mordor.home:8084'.)

== CMD\_DESCRIPTION\_REVERT ==

Reverts an item to a previous revision.

== CMD\_USAGE\_REVERT ==

Usage:

cm ^revert <revspec>

revspec Specification of the changeset that contains the

revision which content will be loaded in the workspace.

(Use 'cm ^help ^objectspec' to learn more about revision

specs.)

== CMD\_HELP\_REVERT ==

Remarks:

The item must be checked in.

Examples:

cm ^revert dir#^cs:0

cm ^revert C:\mywks\dir\file1.txt#23456

== CMD\_DESCRIPTION\_REVISION\_HISTORY ==

Displays the history of a file or directory.

== CMD\_USAGE\_REVISION\_HISTORY ==

Usage:

cm ^history | ^hist <item\_path>[ ...] [--^long | --^format=<str\_format>]

[--^symlink] [--^xml[=<output\_file>]] [--^encoding=<name>]

item\_path Items path. Use a whitespace to separate paths. Use

double quotes (" ") to specify paths containing spaces.

Paths can be server path revisions too.

(Use 'cm ^help ^objectspec' to learn more about specs.)

Options:

--^long Shows additional information.

--^format Retrieves the output message in a specific format. See

Remarks for more info.

--^symlink Applies the history operation to the symlink and not to

the target.

--^xml Prints the output in XML format to the standard output.

It is possible to specify an output file.

--^encoding Used with the '--^xml' option, specifies the encoding to

use in the XML output, i.e.: utf-8.

See the MSDN documentation at

http://msdn.microsoft.com/en-us/library/system.text.encoding.aspx

to get the table of supported encodings and its format,

(at the end of the page, in the "Name" column).

== CMD\_HELP\_REVISION\_HISTORY ==

Remarks:

This command shows a list of revisions for a given item, and label, branch,

and comment info for each revision.

This command accepts a format string to show the output.

The output parameters of this command are the following:

{0} | {^date} Date.

{1} | {^changesetid} Changeset number.

{2} | {^branch} Branch.

{4} | {^comment} Comment.

{5} | {^owner} Owner.

{6} | {^id} Revision id.

{7} | {^repository} Repository.

{8} | {^server} Server.

{9} | {^repspec} Repository spec.

{^tab} Inserts a tab space.

{^newline} Inserts a new line.

Examples:

cm ^history file1.txt "file 2.txt"

cm ^hist c:\workspace --^long

(Displays all information.)

cm ^history link --^symlink

(Applies the history operation to the 'link' file and not to the target,

available on UNIX environments.)

cm ^history ^serverpath:/src/foo/bar.c#^br:/main/task001@myserver

(Retrieves the revision history from a server path in a given branch.)

== CMD\_DESCRIPTION\_REVISION\_TREE ==

Shows the revision tree for an item.

== CMD\_USAGE\_REVISION\_TREE ==

Usage:

cm ^tree <path> [--^symlink]

path Item path.

Options:

--^symlink Applies the operation to the link file and not to the target.

== CMD\_HELP\_REVISION\_TREE ==

Examples:

cm ^tree fichero1.txt

cm ^tree c:\workspace

cm ^tree link --^symlink

(Applies the operation to the link file and not to the target; available on

UNIX environments.)

== CMD\_DESCRIPTION\_RM ==

Allows the user to delete files and directories.

== CMD\_USAGE\_RM ==

Usage:

cm ^remove | ^rm <command> [options]

Commands:

^controlled (optional)

^private

To get more information about each command run:

cm ^remove <command> --^usage

cm ^remove <command> --^help

== CMD\_HELP\_RM ==

Examples:

cm ^remove \path\controlled\_file.txt

cm ^remove ^private \path\private\_file.txt

== CMD\_DESCRIPTION\_RM\_CONTROLLED ==

Deletes a file or directory from version control.

== CMD\_USAGE\_RM\_CONTROLLED ==

Usage:

cm ^remove | ^rm <item\_path>[ ...] [--^format=<str\_format>]

[--^errorformat=<str\_format>] [--^nodisk]

item\_path Items path to remove. Use double quotes (" ") to specify

paths containing spaces. Use a whitespace to separate

paths.

Options:

--^format Retrieves the output progress message in a specific

format. See the Examples for more information.

--^errorformat Retrieves the error message (if any) in a specific

format. See the Examples for more information.

--^nodisk Removes from version control, but keeps the item on

disk.

== CMD\_HELP\_RM\_CONTROLLED ==

Remarks:

Items are deleted from disk. Removed items are removed from the parent

directory in the source code control.

Requirements:

- The item must be under source code control.

Reading input from stdin:

The '^remove' command can read paths from stdin. To do this, pass a single

dash "-".

Example: cm ^remove -

Paths will be read until an empty line is entered.

This allows you to use pipe to specify which files to remove.

Example:

dir /S /B \*.c | cm ^remove -

(In Windows, removes all .c files in the workspace.)

Examples:

cm ^remove src

(Removes 'src'. If src is a directory, this is the same as

'cm ^remove -^R src'.)

cm ^remove c:\workspace\file.txt --^format="{0} - REMOVED" \

--^errorformat="{0} - ERROR REMOVING"

(Removes 'file.txt' from version control and from disk, writing

"c:\workspace\file.txt - ^REMOVED" if the operation succeeded, or

"c:\workspace\file.txt - ^ERROR ^REMOVING" otherwise.)

cm ^remove c:\workspace\file.txt --^nodisk

(Removes 'file.txt' from version control, but keeps it on disk.)

== CMD\_DESCRIPTION\_RM\_PRIVATE ==

Deletes a private file or directory.

Warning: files deleted using the command are permanently erased, and are not

recoverable. It is recommended that you use the '--^dry-run' option to check

which files will be affected by the command.

== CMD\_USAGE\_RM\_PRIVATE ==

Usage:

cm ^remove | ^rm ^private <path>[ ...] [-^R | -^r | --^recursive] [--^ignored]

[--^verbose] [--^dry-run]

path Path of the files or directories to remove. Use double

quotes (" ") to specify paths containing spaces. Use a

whitespace to separate paths.

Options:

--^r Recursively deletes private files from within controlled

directories.

--^ignored Deletes also ignored and cloaked files and directories.

--^verbose Prints all affected paths.

--^dry-run Runs the command without making any changes on disk.

== CMD\_HELP\_RM\_PRIVATE ==

Remarks:

If the path is a private file or directory, it will be deleted from disk.

If the path is a controlled file, the command fails.

If the path is a controlled directory, the command fails unless you

specify the '-^r' option, in which case it will delete all private files and

directories below the specified directory.

Examples:

cm ^remove ^private private\_directory

(Deletes 'private\_directory'.)

cm ^remove ^private c:\workspace\controlled\_directory

(Fails, because 'controlled\_directory' is not private.)

cm ^remove ^private -^r c:\workspace\controlled\_directory

(Deletes all private files and directories below 'controlled\_directory'.)

cm ^rm ^private --^dry-run --^verbose c:\workspace\controlled\_directory -^r

(Shows all of the paths affected by the deletion of private files below

'controlled\_directory' without actually deleting anything.)

cm ^rm ^private --^verbose c:\workspace\controlled\_directory -^r

(Shows all of the paths affected by the deletion of private files below

'controlled\_directory', performing the delete.)

== CMD\_DESCRIPTION\_TRIGGER\_DELETE ==

Deletes a trigger.

== CMD\_USAGE\_TRIGGER\_DELETE ==

Usage:

cm ^trigger | ^tr ^delete | ^rm <subtype-type> <position\_number>

[--^server=<repserverspec>]

subtype-type Trigger execution and trigger operation.

Type 'cm ^showtriggertypes' to see a list of trigger

types.

position\_number Position assigned to the trigger when it was created.

Options:

--^server Deletes the trigger on the specified server.

If no server is specified, executes the command on the

one configured on the client.

== CMD\_HELP\_TRIGGER\_DELETE ==

Examples:

cm ^trigger ^delete ^after-setselector 4

cm ^tr ^rm ^after-setselector 4

== CMD\_DESCRIPTION\_ATTRIBUTE\_SET ==

Sets an attribute on a given object.

== CMD\_USAGE\_ATTRIBUTE\_SET ==

Usage:

cm ^attribute | ^att ^set <att\_spec> <object\_spec> <att\_value>

att\_spec Attribute specification. (Use 'cm ^help ^objectspec' to

learn more about attribute specs.)

object\_spec Specification of the object to set the attribute on.

Attributes can be set on: branches, changesets,

shelvesets, labels, items, and revisions.

(Use 'cm ^help ^objectspec' to learn more about specs.)

att\_value The attribute value to set to the object.

== CMD\_HELP\_ATTRIBUTE\_SET ==

Remarks:

An attribute can be set on an object to save additional information for

this object.

Attributes can be set on the following objects: branches, changesets,

shelvesets, labels, items, and revisions.

Examples:

cm ^attribute ^set ^att:status ^br:/main/SCM105 open

(Sets attribute 'status' to branch 'SCM105' with value 'open'.)

cm ^att ^set ^att:integrated@reptest@server2:8084 ^lb:LB008@reptest@server2:8084 yes

(Sets attribute 'integrated' to label 'LB008' in repository 'reptest' with

value 'yes'.)

== CMD\_DESCRIPTION\_SETOWNER ==

Sets the owner of an object.

== CMD\_USAGE\_SETOWNER ==

Usage:

cm ^setowner | ^sto --^user=<usr\_name> | --^group=<group> <object\_spec>

--^user User name. New owner of the object.

--^group Group name. New owner of the object.

object\_spec Specification of the object to set the new owner on.

The owner can be set on the following objects:

repository server, repository, branch, changeset,

label, item, revision and attribute.

(Use 'cm ^help ^objectspec' to learn more about specs.)

== CMD\_HELP\_SETOWNER ==

Remarks:

The owner of an object can be a user or a group.

The owner can be set on the following objects: repository server,

repository, branch, changeset, label, item, revision, and attribute.

Examples:

cm ^setowner --^user=john ^repserver:localhost:8084

(Sets 'john' as repository server owner.)

cm ^sto --^group=development ^rep:mainRep@PlasticServer:8084

(Sets 'development' group as owner of 'mainRep' repository.)

== CMD\_DESCRIPTION\_SETSELECTOR ==

Sets the selector to a workspace.

== CMD\_USAGE\_SETSELECTOR ==

This command is deprecated. It is still present for backwards compatibility

but selectors were largely deprecated in Plastic SCM 4.0. Selectors still

exist to specify the working branch or changeset, but the old rules to

filter paths are no longer supported.

Usage:

cm ^setselector | ^sts [--^file=<selector\_file>] [--^ignorechanges]

[--^forcedetailedprogress] [<wk\_path> | <wk\_spec>]

Options:

--^file File to load a selector from.

--^ignorechanges Ignores the pending changes warning message that is

shown if there are pending changes detected when

updating the workspace.

--^forcedetailedprogress Forces detailed progress even when standard output

is redirected.

wk\_path Path of the workspace to set the selector.

wk\_spec Workspace specification. (Use 'cm ^help ^objectspec'

to learn more about workspace specs.)

== CMD\_HELP\_SETSELECTOR ==

Remarks:

This command sets the selector of a workspace.

A workspace needs information to load revisions from the repository.

To get this information, Plastic SCM uses a selector.

Using a selector, it is possible to load revisions from a given branch,

label, or changeset.

If a file to load the selector is not specified, the default Operating

System editor will be executed.

Sample selector:

^repository "^default" // working repository

^path "/" // rules will be applied to the root directory

^branch "/^main" // obtain latest revisions from ^br:/^main

^checkout "/^main" // place checkouts on branch ^br:/^main

Examples:

cm ^sts

(Opens the current selector file to be applied.)

cm ^sts ^wk:workspace\_projA@reptest

(Opens the specified selector file to be applied.)

cm ^setselector --^file=c:\selectors\sel.xml

(Sets the specified selector file in the current workspace.)

cm ^setselector --^file=c:\selectors\sel.xml ^wk:MyWorkspace

(Sets the specified selector file in the selected workspace.)

== CMD\_DESCRIPTION\_SHELVE ==

Shelves the contents of checked-out items.

== CMD\_USAGE\_SHELVE ==

This command is deprecated. Use 'cm ^shelveset' instead.

Usage:

cm ^shelve [<item\_path>+] [--^all] [--^dependencies]

[-^c=str\_comment | -^commentsfile=<comments\_file>]

[--^encoding=name] [--^comparisonmethod=comp\_method]

(Shelves the contents.)

cm ^shelve --^apply=<sh\_spec> [--^mount]

(Applies a stored shelveset.)

--^apply Restores the shelved contents of the specified shelveset.

Shelve specification: check 'cm ^help ^objectspec'.

cm ^shelve --^delete=<sh\_spec>

(Removes a stored shelveset.)

--^delete Removes the specified shelveset.

Shelveset specification: check 'cm ^help ^objectspec'.

Options:

item\_path Items to be shelved, separated by spaces. Quotes (") can

be used to specify paths containing spaces.

--^all The items changed, moved and deleted locally, on the

given paths, will also be included.

--^dependencies Includes local change dependencies in the items to

shelve.

-^c Applies the specified comment to the created shelveset.

-^commentsfile Applies the comment in the specified file to the created

shelveset.

--^encoding Specifies the output encoding, i.e.: utf-8.

See the MSDN documentation at

http://msdn.microsoft.com/en-us/library/system.text.encoding.aspx

to get the table of supported encodings and its format,

(at the end of the page, in the "Name" column).

--^comparisonmethod Sets the comparison method. See remarks for more info.

--^mount The mount point for the given repository.

== CMD\_HELP\_SHELVE ==

Remarks:

If neither <item\_path> nor any option is specified, the shelve will involve

all the pending changes in the workspace.

The shelve operation is always applied recursively from the given path.

Requirements to shelve an item:

- The item must be under source code control.

- The item must be checked out or changed (--^all option must be used).

Comparison methods:

^ignoreeol Ignores end of line differences.

^ignorewhitespaces Ignores whitespace differences.

^ignoreeolwhitespaces Ignores end of line and whitespace differences.

^notignore Detects end of line and whitespace differences.

Set PLASTICEDITOR environment variable to specify an editor to type the

comment.

Examples:

cm ^shelve -^c="my comment"

(Shelves all the pending changes in the current workspace including a

comment.)

cm ^shelve file1.txt "file 2.txt" -^commentsfile=commentshelve.txt

(Shelves the selected pending changes and applies the comment in the

commentshelve.txt file.)

cm ^shelve --^apply=^sh:3

(Applies a stored shelveset.)

cm ^shelve --^delete=^sh:3

(Removes a stored shelveset.)

cm ^status --^short --^changelist=pending\_to\_review | cm ^shelve -

(Shelves client changelist.

The command above lists the paths in the changelist named

'pending\_to\_review' and the path list is redirected to the input of the

shelve command.)

== CMD\_DESCRIPTION\_SHELVESET ==

Allows the user to manage shelvesets.

== CMD\_USAGE\_SHELVESET ==

Usage:

cm ^shelveset <command> [options]

Commands:

^create | ^mk

^delete | ^rm

^apply

To get more information about each command run:

cm ^shelveset <command> --^usage

cm ^shelveset <command> --^help

== CMD\_HELP\_SHELVESET ==

Examples:

cm ^shelveset ^create -^c="my comment"

cm ^shelveset ^delete ^sh:3

cm ^shelve ^apply ^sh:3

== CMD\_DESCRIPTION\_SHELVESET\_CREATE ==

Shelves pending changes.

== CMD\_USAGE\_SHELVESET\_CREATE ==

Usage:

cm ^shelveset ^create | ^mk [<item\_path>[ ...]] [--^all] [--^dependencies]

[-^c=<str\_comment> | -^commentsfile=<comments\_file>]

Options:

item\_path Items to shelve. Use a whitespace to separate user names.

Use double quotes (" ") to specify paths containing

spaces.

--^all The items changed, moved, and deleted locally, on the

given paths, will also be included.

--^dependencies Includes local change dependencies into the items to

shelve.

-^c Applies the specified comment to the created shelve.

-^commentsfile Applies the comment in the specified file to the created

shelve.

== CMD\_HELP\_SHELVESET\_CREATE ==

The '^shelveset ^create' command stores the contents of checked out items inside the

repository. This way the contents are protected without the need to

checkin the files.

Remarks:

If neither <item\_path> nor any option is specified, the shelveset will

include all the pending changes in the workspace.

The '^shelveset ^create' operation is always applied recursively from the

given path.

Requirements to shelve an item:

- The item must be under source code control.

- The item must be checked out or changed ('--^all' option must be used).

Set PLASTICEDITOR environment variable to specify an editor to type the

comment.

Examples:

cm ^shelveset ^create -^c="my comment"

(Shelves all the pending changes in the current workspace including a

comment.)

cm ^shelveset file1.txt "file 2.txt" -^commentsfile=commentshelve.txt

(Shelves the selected pending changes and applies the comment in the

'commentshelve.txt' file. Note, '^create' is the default subcommand.)

cm ^status --^short --^changelist=pending\_to\_review | cm ^shelveset -

(Shelves client changelist.

The command above lists the paths in the changelist named

'pending\_to\_review' and the path list is redirected to the input of the

'^shelveset' command.)

== CMD\_DESCRIPTION\_SHELVESET\_DELETE ==

Deletes a shelveset.

== CMD\_USAGE\_SHELVESET\_DELETE ==

Usage:

cm ^shelveset ^delete | ^rm <sh\_spec>

sh\_spec Shelveset specification. (Use 'cm ^help ^objectspec' to

learn more about shelveset specs.)

== CMD\_HELP\_SHELVESET\_DELETE ==

The '^shelveset ^delete' command deletes a shelveset.

Examples:

cm ^shelveset ^delete ^sh:3

(Removes a stored shelveset.)

== CMD\_DESCRIPTION\_SHELVESET\_APPLY ==

Applies a stored shelveset.

== CMD\_USAGE\_SHELVESET\_APPLY ==

Usage:

cm ^shelveset ^apply <sh\_spec> [--^mount] [--^encoding=<name>]

[--^comparisonmethod=(^ignoreeol | ^ignorewhitespaces| \

^ignoreeolwhitespaces | ^notignore)]

sh\_spec Shelveset specification. (Use 'cm ^help ^objectspec' to

learn more about shelveset specs.)

Options:

--^mount The mount point for the given repository.

--^encoding Specifies the output encoding, i.e.: utf-8.

See the MSDN documentation at

http://msdn.microsoft.com/en-us/library/system.text.encoding.aspx

to get the table of supported encodings and its format,

(at the end of the page, in the "Name" column).

--^comparisonmethod Sets the comparison method. See Remarks for more info.

== CMD\_HELP\_SHELVESET\_APPLY ==

The '^shelveset ^apply' command restores the contents of a stored shelveset.

Remarks:

Comparison methods:

^ignoreeol Ignores end of line differences.

^ignorewhitespaces Ignores whitespace differences.

^ignoreeolwhitespaces Ignores end of line and whitespace differences.

^notignore Detects end of line and whitespace differences.

Examples:

cm ^shelveset ^apply ^sh:3

(Applies a stored shelve.)

== CMD\_DESCRIPTION\_SHOW\_FIND\_OBJECTS ==

Lists objects and attributes.

== CMD\_USAGE\_SHOW\_FIND\_OBJECTS ==

Usage:

cm ^showfindobjects

== CMD\_HELP\_SHOW\_FIND\_OBJECTS ==

Available objects and attributes:

^attribute:

You can find attributes by filtering using the following fields:

^type : string.

Example:

cm ^find ^attribute "^where ^type = 'status'"

(Finds all attributes of type 'status'.)

^value : string.

^date : date.

Check "date constants" for more info in this guide.

Example:

cm ^find ^attribute "^where ^date > '^this ^week'"

(Finds all attributes applied during the current week.)

^owner : user.

Admits special user '^me'.

Example:

cm ^find ^attribute "^where ^value = 'resolved' ^and ^owner = '^me'"

(Finds all attributes with value 'resolved' applied by me.)

^GUID : Global Unique Identifier.

Hexadecimal id in the format xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx.

^comment : string.

^srcobj : object spec: item path, branch, changeset, revision, or label.

Use 'cm ^help ^objectspec' to learn how to specify these objects.

Examples:

cm ^find ^attribute "^where ^srcobj = '^item:readme.txt'"

(Finds the attributes applied to the item 'readme.txt'.)

cm ^find ^attribute "^where ^srcobj = '^br:/main/scm23343'"

(Finds the attributes applied to the branch scm23343.)

cm ^find ^attribute "^where ^srcobj = '^rev:readme.txt#^br:/main/task002'"

(Finds the attributes applied to the specified revision.)

cm ^find ^attribute "^where ^srcobj = '^rev:^revid:1126'"

(Finds the attributes applied to the specified revision id.)

^ID : integer.

^attributetype:

You can find attribute types by filtering using the following fields:

^name : string.

Example:

cm ^find ^attributetype "^where ^name ^like 'st%'"

(Finds all attribute where name starts with 'st'.)

^value : string.

^date : date.

Check "date constants" for more info in this guide.

Example:

cm ^find ^attribute "^where ^date > '^today'"

(Finds all attributes applied today.)

^owner : user.

Admits special user '^me'.

^GUID : Global Unique Identifier.

Hexadecimal id in the format xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx.

^comment : string.

Example:

cm ^find ^attributetype "^where ^comment != ''" --^xml

(Finds all attribute types that have a comment and prints the

output in XML format to the standard output.)

^source : object spec: item path, branch, changeset or label.

Use 'cm ^help ^objectspec' to learn how to specify these objects.

Example:

cm ^find ^attributetype "^where ^source = '^item:readme.txt'"

(Finds all attribute types in item 'readme.txt'.)

cm ^find ^attributetype "^where ^source = '^cs:30'"

(Finds all attribute types in changeset '30'.)

cm ^find ^attributetype "^where ^source = '^lb:v0.14.1'"

(Finds all attribute types in label 'v0.14.1'.)

^ID : integer.

Replication field. Check "replication related fields" below.

^ReplLogId

^ReplSrcDate

^ReplSrcId

^ReplSrcRepository

^ReplSrcServer

^branch:

You can find branches by filtering using the following fields:

^name : string.

Example:

cm ^find ^branch "^where ^name ^like 'scm23%'"

(Finds branches which name starts with 'scm23'.)

^date : date.

Check "date constants" for more info in this guide.

Example:

cm ^find ^branch "^where ^date > '^one ^week ^ago'"

(Finds branches created during the last week.)

^changesets : date (of the changesets in the branch).

Check "date constants" for more info in this guide.

Example:

cm ^find ^branch "^where ^changesets >= '^today'"

(Finds branches with changesets created today.)

^attribute : string.

^attrvalue : string.

Example:

cm ^find ^branch "^where ^attribute = 'status' ^and ^attrvalue = 'failed'"

(Finds branches that have the attribute 'status' and which

value is 'failed'.)

^owner : user.

Admits special user '^me'.

^parent : branch spec.

Use 'cm ^help ^objectspec' to learn how to specify this object.

Example:

cm ^find ^branch "^where ^owner != '^me' ^and ^parent != '^br:/main'"

(Finds branches created by other than me and which parent

branch is not '/main'.)

^comment : string.

^GUID : Global Unique Identifier.

Hexadecimal id in the format xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx.

^ID : integer.

Example:

cm ^find ^branch "^where ^id = 2029607"

(Finds the branch which id is 2029607.)

Replication fields. Check "replication related fields" below.

^ReplLogId

^ReplSrcDate

^ReplSrcId

^ReplSrcRepository

^ReplSrcServer

^changeset:

You can find changesets by filtering using the following fields:

^branch : branch spec.

Use 'cm ^help ^objectspec' to learn how to specify this

object.

Example:

cm ^find ^changeset "^where ^branch = '/main/scm23119'"

(Finds all changesets in branch 'scm23119'.)

^changesetid : integer.

^attribute : string.

Example:

cm ^find ^changeset "^where ^attribute = 'status'"

(Finds the changesets with the attribute 'status'.)

^attrvalue : string.

^date : date.

Check "date constants" for more info in this guide.

^owner : user.

Admits special user '^me'.

Example:

cm ^find ^changeset "^where ^date >= '6/8/2018' ^and ^owner != '^me'"

(Finds all changesets with creation date equal or

greater than 6/8/2018 and created by others than me.)

^GUID : Global Unique Identifier.

Hexadecimal id in the format xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx.

Example:

cm ^find ^changeset "^where ^guid = '1b30674f-14cc-4fd7-962b-676c8a6f5cb6'"

(Finds the changeset with the specified guid.)

^comment : string.

Example:

cm ^find ^changeset "^where ^comment = ''"

(Finds the changesets with no comments.)

^onlywithrevisions : boolean.

To filter whether a cset has revisions or not.

Example:

cm ^find ^changeset "^where ^onlywithrevisions = 'false'"

(Finds changesets with no revisions.)

^returnparent : boolean.

A way to return the parent of a cset. Good for scripting.

Example:

cm ^find ^changeset "^where ^changesetid = 29 ^and ^returnparent = 'true'"

(Finds the parent of changeset 29.)

^parent : changeset id (integer).

Example:

cm ^find ^changeset "^where ^parent = 548"

(Finds all changesets which parent is cset 548.)

^ID : integer.

Replication fields. Check "replication related fields" below.

^ReplLogId

^ReplSrcDate

^ReplSrcId

^ReplSrcRepository

^ReplSrcServer

^label:

You can find labels by filtering using the following fields:

^name : string.

Example:

cm ^find ^label "^where ^name ^like '7.0.16.%'"

(Finds the labels with a name that starts with '7.0.16.'.)

^attribute : string.

^attrvalue : string.

^date : date.

Check "date constants" for more info in this guide.

Example:

cm ^find ^label "^where ^date >= '^this ^month' ^and \

^attribute = 'publish-status' ^and ^attrvalue != 'PUBLISHED'"

(Finds the labels created this month with an attribute 'publish-status'

set to a value other than 'PUBLISHED'.)

^owner : user.

Admits special user '^me'.

^GUID : Global Unique Identifier.

Hexadecimal id in the format xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx.

^branch : branch spec.

Use 'cm ^help ^objectspec' to learn how to specify this object.

Example:

cm ^find ^label "^where ^branch = '/main'"

(Finds all labels applied to the main branch.)

^branchid : integer.

^changeset : changeset id (integer).

Example:

cm ^find ^label "^where ^changeset = 111733"

(Finds the labels applied to changeset 111733.)

^comment : string.

^ID : integer.

Replication fields. Check "replication related fields" below.

^ReplLogId

^ReplSrcDate

^ReplSrcId

^ReplSrcRepository

^ReplSrcServer

^merge:

You can find merges by filtering using the following fields:

^srcbranch : branch spec.

Use 'cm ^help ^objectspec' to learn how to specify this object.

Example:

cm ^find ^merge "^where ^srcbranch = '^br:/main'"

(Finds merges from the main branch.)

^srcchangeset : changeset id (integer).

^dstbranch : branch spec.

Use 'cm ^help ^objectspec' to learn how to specify this object.

^dstchangeset : changeset id (integer).

Example:

cm ^find ^merge "^where ^dstchangeset = 108261" \

--^format="{^srcbranch} {^srcchangeset} {^dstbranch} {^dstchangeset} {^owner}"

(Finds the merges to changeset 108261 and prints the

formatted output showing the source (branch and cset id),

the destination (branch and cset id), and the merge owner.)

^date : date.

Check "date constants" for more info in this guide.

^owner : user.

Admits special user '^me'.

^GUID : Global Unique Identifier.

Hexadecimal id in the format xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx.

^type : string.

Possible values are '^merge', '^cherrypick',

'^cherrypicksubstractive', '^interval', '^intervalcherrypick'

and '^intervalcherrypicksubstractive'

Example:

cm ^find ^merge "^where ^type = '^cherrypick' ^and ^owner = '^me'"

(Finds all my cherrypicks.)

^ID : integer.

^replicationlog:

You can find replication log by filtering using the following fields:

^branch : branch spec.

Use 'cm ^help ^objectspec' to learn how to specify this object.

Example:

cm ^find ^replicationlog "^where ^branch = '/main/gm22358'"

(Finds the replication logs of branch 'gm22358'.)

^repositoryname : string.

^owner : user.

Admits special user '^me'.

^date : date.

Check "date constants" for more info in this guide.

^server : string.

^package : boolean.

Example:

cm ^find ^replicationlog "^where ^package = 'T' ^and ^server ^like '%cloud%'"

(Finds the replication logs created from package which

server name contains 'cloud'.)

^ID : integer.

^review:

You can find code reviews by filtering using the following fields:

^status : string.

^assignee : string.

Example:

cm ^find ^review "^where ^status = 'pending' ^and ^assignee = '^me'"

(Finds all my pending reviews.)

^title : string.

^target : object spec: branch or changeset.

Use 'cm ^help ^objectspec' to learn how to specify this object.

Example:

cm ^find ^review "^where ^target = '^br:/main/scm17932'"

(Finds the reviews related to branch 'scm17932'.)

^targetid : integer.

^targettype : string.

Possible values are '^branch' and '^changeset'.

Example:

cm ^find ^review "^where ^targettype = '^changeset'"

(Finds the reviews which target type is changeset.)

^date : date.

Check "date constants" for more info in this guide.

^owner : user.

Admits special user '^me'.

^GUID : Global Unique Identifier.

Hexadecimal id in the format xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx.

^ID : integer.

^revision:

You can find revisions by filtering using the following fields:

^branch : branch spec.

Use 'cm ^help ^objectspec' to learn how to specify this object.

^changeset : changeset id (integer).

Example:

cm ^find ^revision "^where ^changeset >= 111756"

(Finds the revisions created in changeset 111756

and later.)

^item : string or integer.

^itemid : integer.

Examples:

cm ^find ^revision "^where ^item = 'readme.txt' ^or ^itemid = 2250"

(Finds the revisions of item 'readme.txt' plus

item id 2250.)

cm ^find ^revision "^where ^item = 'readme.txt' ^or ^item = 2250"

(Gets the same revisions as the previous example.)

^attribute : string.

^attrvalue : string.

Example:

cm ^find ^revision "^where ^attribute = 'status' ^and ^attrvalue != 'open'"

(Finds the revisions with attribute 'status' which

value is other than 'open'.)

^archived : boolean.

Example:

cm ^find ^revision "^where ^archived = 'true'"

(Finds the revisions that are archived in an

external storage.)

^comment : string.

^date : date.

Check "date constants" for more info in this guide.

^GUID : Global Unique Identifier.

Hexadecimal id in the format xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx.

^owner : user.

Admits special user '^me'.

^parent : revision id (integer).

^returnparent : boolean.

^shelve : shelve id (integer).

^size : integer (in bytes).

^type : string.

Possible values are '^dir', '^bin', and '^txt'.

Example:

cm ^find ^revision "^where ^type = '^txt' and \

^size > 300000 ^and ^owner = '^me' and ^date >= '2 ^months ^ago'"

(Finds the text revisions created by me two months

ago and with size greater than about 3MB.)

^workspacecheckoutid : integer.

^ID : integer.

Replication fields. Check "replication related fields" below.

^ReplLogId

^ReplSrcDate

^ReplSrcId

^ReplSrcRepository

^ReplSrcServer

^shelve:

You can find shelves by filtering using the following fields:

^owner : user.

Admits special user '^me'.

^date : date.

Check "date constants" for more info in this guide.

Example:

cm ^find ^shelve "^where ^owner != '^me' ^and ^date >= '^1 ^years ^ago'"

(Finds the shelves created by others than me during the last

year.)

^attribute : string.

^attrvalue : string.

^comment : string.

^GUID : Global Unique Identifier.

Hexadecimal id in the format xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx.

^parent : integer.

^shelveid : integer.

Example:

cm ^find ^shelve "^where ^shelveid = 2"

(Finds the shelve with name 2.)

^ID : integer.

Example:

cm ^find ^shelve "^where ^id >= 3848"

(Finds the shelves which object id is greater than 3848.)

Replication fields. Check "replication related fields" below.

^ReplLogId

^ReplSrcDate

^ReplSrcId

^ReplSrcRepository

^ReplSrcServer

Replication related fields:

Many objects track replication data, meaning Plastic tracks where they were

originally created.

The fields you can use are:

^ReplSrcServer : repspec. Stands for "replication source server".

Server where the object was replicated from.

Example:

cm ^find ^branch "^where ^replsrcserver='skull.codicefactory.com:9095'"

(Finds the branches replicated from server 'skull'.)

^ReplSrcRepository : string. Stands for "replication source repo". It is

the repository where the object was replicated from.

Example:

cm ^find ^branch "^where ^replsrcserver = 'skull.codicefactory.com:9095' \

^and ^replsrcrepository = 'codice'"

(Finds the branches replicated from server 'skull'

and from repository 'codice'.)

^ReplLogId : integer. ID of the replication operation. In Plastic,

each time new objects are created from a replica,

a new 'replicationlog' is created.

Example:

cm ^find ^revision "^where ^repllogid = 2019974"

(Finds the revisions replicated from replica

2019974.)

^ReplSrcDate : date. It is the date when the replica actually took

place.

Replicated objects will retain its original creation

date, o this field is useful if you want to find

objects that where replicated within a specific

timeframe.

Example:

cm ^find ^label "^where ^replsrcdate >= '^one ^month ^ago' \

^and ^date >= '15 ^days ^ago'"

(Finds the labels created 15 days ago and were

replicated one month ago.)

cm ^find ^replicationlog "^where ^date > '^one ^week ^ago'"

8780433 27/09/2018 8:49:38 codice@BACKYARD:8087 F mbarriosc

(Finds the replication logs created one week ago.)

Now, you can check that the replicated branch was

created before it was replicated over:

cm ^find ^branch "^where ^repllogid = 8780433"

8780443 26/09/2018 12:20:55 /main/scm23078 maria codice T

^ReplSrcId : integer. It is the ID of the replication source server.

You can discover this ID searching for

'^replicationsource' objects with the 'cm ^find' command.

Example:

cm ^find ^replicationsource

7860739 codice@AFRODITA:8087 d9c4372a-dc55-4fdc-ad3d-baeb2e975f27

8175854 codice@BACKYARD:8087 66700d3a-036b-4b9a-a26f-adfc336b14f9

Now, you can find the changesets replicated from

codice@AFRODITA:8087:

cm ^find ^changesets "^where ^replsrcid = 7860739"

Date constants:

You can use date formats that follow your machine localization settings.

For example, if your computer displays dates in the format 'MM-dd-yyyy',

you can use dates such as '12-31-2019' in your queries.

You can also use the following constants to simplify your queries:

'^today' : today's date.

'^yesterday' : yesterday's date.

'^this ^week' : current week's Monday date.

'^this ^month' : current month's 1st day date.

'^this ^year' : current year's January 1st date.

'^one ^day ^ago' : one day before the current date.

'^one ^week ^ago' : seven days before the current date.

'^one ^month ^ago' : one month before the current date.

'n ^days ^ago' : 'n' days before the current date.

'n ^months ^ago' : 'n' months before the current date.

'n ^years ^ago' : 'n' years before the current date.

The following '^where' clauses are valid for fields of type '^date':

'(...) ^where ^date > '^today' (...)'

'(...) ^where ^date < '^yesterday' (...)'

'(...) ^where ^date > '^this ^week' (...)'

'(...) ^where ^date > '^this ^month' (...)'

'(...) ^where ^date < '^one ^day ^ago' ^and ^date > '3 ^days ^ago' (...)'

'(...) ^where ^date < '^one ^week ^ago' ^and ^date > '3 ^weeks ^ago' (...)'

'(...) ^where ^date < '^one ^month ^ago' ^and ^date > '3 ^months ^ago' (...)'

'(...) ^where ^date > '1 ^year ^ago' (...)'

You can also force a specific date format on the 'cm ^find' command using the

--^dateformat flag. Check 'cm ^find --^help' for further details.

== CMD\_DESCRIPTION\_TRIGGER\_SHOWTYPES ==

Displays available trigger types.

== CMD\_USAGE\_TRIGGER\_SHOWTYPES ==

Usage:

cm ^trigger ^showtypes

== CMD\_DESCRIPTION\_SHOWACL ==

Shows the ACL of an object.

== CMD\_USAGE\_SHOWACL ==

Usage:

cm ^showacl | ^sa <object\_spec> [--^extended] [--^xml[=<output\_file>]]

[--^encoding=<name>]

object\_spec Specification of the object to show the ACL of.

The valid objects for this command are:

repserver, repository, branch, changeset, label, item,

and attribute.

(Use 'cm ^help ^objectspec' to learn more about specs.)

Options:

--^extended Shows ACL hierarchy tree.

--^xml Prints the output in XML format to the standard output.

It is possible to specify an output file.

--^encoding Used with the '--^xml' option, specifies the encoding to

use in the XML output, i.e.: utf-8.

See the MSDN documentation at

http://msdn.microsoft.com/en-us/library/system.text.encoding.aspx

to get the table of supported encodings and its format,

(at the end of the page, in the "Name" column).

== CMD\_HELP\_SHOWACL ==

Examples:

cm ^showacl ^repserver:PlasticServer:8084

(Shows the ACL of the selected server.)

cm ^sa ^br:/main --^extended

(Shows the ACL hierarchy tree of the selected branch specification.)

== CMD\_DESCRIPTION\_SHOWCOMMANDS ==

Shows all the available commands.

== CMD\_USAGE\_SHOWCOMMANDS ==

Usage:

cm ^showcommands

== CMD\_HELP\_SHOWCOMMANDS ==

== CMD\_DESCRIPTION\_SHOWOWNER ==

Shows the owner of an object.

== CMD\_USAGE\_SHOWOWNER ==

Usage:

cm ^showowner | ^so <object\_spec>

object\_spec Specification of the object to show the owner of.

The object must be one of the following:

repository server, repository, branch, changeset,

label, attribute, revision, and item.

(Use 'cm ^help ^objectspec' to learn more about specs.)

== CMD\_HELP\_SHOWOWNER ==

Remarks:

This command displays the owner of an object. The owner can be a user or

a group. The owner can be modified with 'cm ^setowner' command.

Examples:

cm ^showowner ^repserver:PlasticServer:8084

(Shows the owner of the selected server.)

cm ^so ^item:samples\

(Shows the owner of the selected item specification.)

== CMD\_DESCRIPTION\_SHOWPERMISSIONS ==

Lists the available permissions.

== CMD\_USAGE\_SHOWPERMISSIONS ==

Usage:

cm ^showpermissions | ^sp

== CMD\_HELP\_SHOWPERMISSIONS ==

Examples:

cm ^showpermissions

== CMD\_DESCRIPTION\_SHOWSELECTOR ==

Shows the workspace selector.

== CMD\_USAGE\_SHOWSELECTOR ==

This command is deprecated. It is still present for backwards compatibility

but selectors were largely deprecated in Plastic SCM 4.0. Selectors still

exist to specify the working branch or changeset, but the old rules to

filter paths are no longer supported.

Usage:

cm ^showselector | ^ss [<wk\_path> | <wk\_spec>]

wk\_path Path of the workspace to show the selector.

wk\_spec Workspace specification. (Use 'cm ^help ^objectspec' to

learn more about workspace specs.)

== CMD\_HELP\_SHOWSELECTOR ==

Remarks:

If neither path nor workspace spec is specified, the command will take the

current directory as the workspace path.

Examples:

cm ^showselector c:\workspace

(Shows the selector for the selected workspace path.)

cm ^ss

(Shows the selector for current workspace.)

cm ^showselector > mySelector.txt

(Writes into a file the selector for the current workspace.)

cm ^showselector ^wk:mywk@reptest

(Shows the selector for the workspace 'mywk' in the repository 'reptest'.)

== CMD\_DESCRIPTION\_SUPPORT ==

Allows the user to perform support related operations.

== CMD\_USAGE\_SUPPORT ==

Usage:

cm ^support <command> [options]

Commands:

^bundle

To get more information about each command run:

cm ^support <command> --^usage

cm ^support <command> --^help

== CMD\_HELP\_SUPPORT ==

Examples:

cm ^support

cm ^support ^bundle

cm ^support ^bundle c:\outputfile.zip

== CMD\_DESCRIPTION\_SUPPORT\_BUNDLE ==

Creates a "support bundle" package with relevant logs.

You can attach the file while requesting help, asking for extra info, or

submitting a bug.

== CMD\_USAGE\_SUPPORT\_BUNDLE ==

Usage:

cm ^support ^bundle [<outputfile>]

Options:

outputfile Creates the "support bundle" package at the specified

location.

== CMD\_HELP\_SUPPORT\_BUNDLE ==

Remarks:

This command allows users to create a "support bundle" package which can be

attached when requesting help, asking for extra info, or submitting a bug.

The user can optionally specify a location for the output file; otherwise, the

output file will be written to the temp directory.

Examples:

cm ^support ^bundle

(Creates "support bundle" in temp directory.)

cm ^support ^bundle c:\outputfile.zip

(Creates "support bundle" at the specified location.)

== CMD\_DESCRIPTION\_SWITCH ==

Switches the workspace to a branch, changeset, label, or shelveset.

== CMD\_USAGE\_SWITCH ==

Usage:

cm ^switch (<brspec> | <csetspec> | <lbspec> | <shspec>)

[--^workspace=<path>] [--^repository=<name>]

[--^forcedetailedprogress]

(Use 'cm ^help ^objectspec' to learn more about branch, changeset, label,

and shelveset specifications.)

Options:

--^workspace Path where the workspace is located.

--^repository Switches to the specified repository.

--^forcedetailedprogress Forces detailed progress even when standard

output is redirected.

== CMD\_HELP\_SWITCH ==

Remarks:

This command allows users to update the workspace tree to the contents

of the specified object (branch, label, shelveset, or changeset).

Examples:

cm ^switch ^br:/main

cm ^switch ^lb:Rel1.1

cm ^switch ^br:/main/scm002 --^repository=rep2

cm ^switch ^cs:4375

cm ^switch ^sh:2

== CMD\_DESCRIPTION\_SWITCH\_TO\_BRANCH ==

Sets a branch as the working branch.

== CMD\_USAGE\_SWITCH\_TO\_BRANCH ==

This command is deprecated. Use cm switch instead.

Usage:

cm ^switchtobranch [options] [branch\_spec]

branch\_spec: Branch specification.

Options:

--^label=name | --^changeset=number: load revisions from the specified

label or changeset. One of these options is required if no branch\_spec is

given.

--^changeset=cset: Switch to the specified changeset.

--^repository=rep: Switch to the specified repository.

--^workspace | -wk=path: path where the workspace is located.

== CMD\_HELP\_SWITCH\_TO\_BRANCH ==

Remarks:

This command allows users to work in a branch.

If no branch\_spec specified, a label or a changeset must be specified.

If no repository is specified, the branch is set to the current repository.

Examples:

cm ^switchtobranch ^br:/main

cm ^switchtobranch ^br:/main/task001

cm ^switchtobranch --^label=BL050

(Read-only configuration. The command loads the contents of the labeled

changeset.)

== CMD\_DESCRIPTION\_SYNC ==

Synchronize with Git.

== CMD\_USAGE\_SYNC ==

Usage:

cm ^synchronize | ^sync <repspec> ^git [<url> [--^user=<usr\_name> --^pwd=<pwd>]]

[(--^txtsimilaritypercent | --^binsimilaritypercent | \

--^dirsimilaritypercent)=<value>]

[--^author] [--^skipgitlfs]

repspec Repository specification. (Use 'cm ^help ^objectspec' t

learn more about repository specs.)

git (Default).

Options:

url Remote repository URL (http(s):// or git:// or a

SSH URL).

--^user User name for the specified URL.

--^pwd Password for the specified URL.

--^txtsimilaritypercent \

--^binsimilaritypercent \

--^dirsimilaritypercent

To detect moved items, the same way as Plastic SCM

GUI does.

--^author Uses name and timestamp values from the git author.

(git committer by default)

--^skipgitlfs Ignores the Git LFS configuration in the

.gitattributes file. It acts like without Git LFS

support.

== CMD\_HELP\_SYNC ==

Remarks:

If the git repository requires user and password, then use '^url', '--^user',

and '--^pwd' options.

If the git repository doesn't require user and password, then use '^url'

option with the first sync operation. With next sync operations, '^url'

option is optional.

To use the SSH protocol to perform the sync, you must have the 'ssh' client

added to the PATH environment variable and properly configured to connect

to the remote host (i.e. private/public keys configured).

Examples:

cm ^sync default@localhost:8087 ^git git://localhost/repository

== CMD\_DESCRIPTION\_TRIGGER ==

Allows the user to manage triggers.

== CMD\_USAGE\_TRIGGER ==

Usage:

cm ^trigger | ^tr <command> [options]

Commands:

^create | ^mk

^delete | ^rm

^edit

^list | ^ls

^showtypes

To get more information about each command run:

cm ^trigger <command> --^usage

cm ^trigger <command> --^help

== CMD\_HELP\_TRIGGER ==

Examples:

cm ^tr ^mk ^before-mklabel new "/path/to/script" --^server=myserver:8084

cm ^tr ^edit ^before-mklabel 7 --^position=4 --^server=myserver:8084

cm ^tr ^ls ^before-mkbranch --^server=myserver:8084

cm ^tr ^rm ^after-setselector 4

cm ^tr ^showtypes

== CMD\_DESCRIPTION\_TUBE ==

Runs commands related to Plastic Tube.

== CMD\_USAGE\_TUBE ==

Usage:

cm ^tube ^config -^u=<user> -^p=<password>

(Configures Plastic SCM to use Plastic Tube with the specified user and

password.)

cm ^tube ^create <remoteuser>

(Creates the tube "remoteuser -> myuser".

The user "myuser" allows to "remoteuser" to connect to "myuser" server.

Connections can be established from "remoteuser" to "myuser".

Only tubes from other users to the current tube user can be created.)

cm ^tube ^remove <remoteuser>

(Removes the tube "remoteuser -> myuser".)

cm ^tube ^local

(Lists the local repositories shared in the local server and the users

that it is shared with.)

cm ^tube ^remote

(Lists the shared remote repositories that are shared with the current

tube user.)

cm ^tube ^share <rep\_spec>[ ...] -^u=<remoteuser> -^a=(^pull | ^push | ^pull,^push)

(Shares the local repository(s) with the remote user and sets the specified

access mode. Use a whitespace to separate repository specs.)

cm ^tube ^unshare <rep\_spec>[ ...] -^u=<remoteuser>

(Unshares the local repository(s) with the remote user. Use a whitespace to

separate repository specs.)

cm ^tube ^connect

(Connects the Plastic SCM server to Plastic Tube.)

cm ^tube ^disconnect

(Disconnects the Plastic SCM server from Plastic Tube.)

cm ^tube ^status

(Shows if the Plastic SCM server is connected to Plastic Tube.)

Options

-^u Tube user (the plasticscm.com user).

-^p User password.

-^a Sets the access mode.

rep\_spec Repository specification. (Use 'cm ^help ^objectspec' to

learn more about repository specs.)

== CMD\_HELP\_TUBE ==

Remarks:

Use the 'cm ^tube' command to manage Plastic Tube.

Examples:

cm ^tube ^config -^u=ruben@codicesoftware.com -^p=rubenpassword

cm ^tube ^create pablo@codicesoftware.com

('pablo@codicesoftware.com' can connect to the current Plastic Tube user)

cm ^tube ^remove pablo@codicesoftware.com

cm ^tube ^local

cm ^tube ^remote

cm ^tube ^share repo@server:8087 -^u=pablo@codicesoftware.com -^a=^pull,^push

cm ^tube ^share repo@server:8087 doc@server:8087 -^u=pablo@codicesoftware.com -^a=^push

cm ^tube ^unshare repo@server:8087 -^u=pablo@codicesoftware.com

cm ^tube ^connect

cm ^tube ^disconnect

cm ^tube ^status

== CMD\_DESCRIPTION\_UNCO ==

Undoes the checkout of an item.

== CMD\_USAGE\_UNCO ==

Usage:

cm ^undocheckout | ^unco <item\_path>[ ...] [-^a | --^all] [--^symlink] [--^silent]

[--^machinereadable [--^startlineseparator=<sep>]

[--^endlineseparator=<sep>] [--^fieldseparator=<sep>]]

item\_path Items to apply the operation. Use a whitespace to separate

paths. Use double quotes (" ") to specify paths

containing spaces.

Use . to apply the operation to current directory.

Options:

-^a | --^all Undoes all of the changes in the specified items. If

the item(s) were checked out, the checkout will be

reverted. If the item(s) were locally modified, the

modifications will be reverted.

--^symlink Applies the undocheckout operation to the link and not

to the target.

--^silent Does not show any output.

--^machinereadable Outputs the result in an easy-to-parse format.

--^startlineseparator Used with the '--^machinereadable' flag,

specifies how the lines should start.

--^endlineseparator Used with the '--^machinereadable' flag,

specifies how the lines should end.

--^fieldseparator Used with the '--^machinereadable' flag,

specifies how the fields should be separated.

== CMD\_HELP\_UNCO ==

Remarks:

If an item is checked out and you do not want to checkin it, you can undo

the checkout using this command. Both files and folders can be unchecked

out. The item will be updated to the state it had before checking it out.

Requirements:

- The item must be under source code control.

- The item must be checked out.

Reading input from stdin:

The '^undocheckout' command can read paths from stdin. To do this, pass a

single dash "-".

Example: cm ^undocheckout ^checkin -

Paths will be read until an empty line is entered.

This allows you to use pipe to specify for which files to undo the checkout.

Example:

dir /S /B \*.c | cm ^undocheckout --^all -

(In Windows, undoes the checkout of all .c files in the workspace.)

Examples:

cm ^undocheckout .

(Undoes checkouts in the current directory.)

cm ^undocheckout file1.txt file2.txt

cm unco c:\workspace\file.txt

(Undoes checkouts of the selected files.)

cm ^unco -^a file1.txt

(Undoes checkouts or local modifications of 'file1.txt')

cm ^unco link --^symlink

(Applies the undocheckout operation to the link file and not to the target,

available on UNIX environments.)

cm ^status --^short --^changelist=pending\_to\_review | cm ^undocheckout -

(Undoes client changelist.

The command above will list the paths in the changelist named

'pending\_to\_review' and the path list will be redirected to the input of the

undocheckout command).

cm ^unco . --^machinereadable

(Undoes checkouts in the current directory, and prints the result in a

simplified, easier-to-parse format.)

cm ^unco . --^machinereadable --^startlineseparator=">" --^endlineseparator="<" \

--^fieldseparator=","

(Undoes checkouts in the current directory, and prints the result in a

simplified, easier to parse format, starting and ending the lines, and

separating the fields, with the specified strings.)

== CMD\_DESCRIPTION\_UNCOUNCHANGED ==

Undoes non-changed checked out items.

== CMD\_USAGE\_UNCOUNCHANGED ==

Usage:

cm ^uncounchanged | ^unuc <item\_path>[ ...] [-^R | -^r | --^recursive]

[--^symlink] [--^silent]

item\_path Items to apply the operation. Use a whitespace to separate

paths. Use double quotes (" ") to specify paths

containing spaces.

Use . to apply the operation to current directory.

Options:

-^R Undoes unchanged items recursively in the specified paths.

--^symlink Applies the uncounchanged operation to the link and not

to the target.

--^silent Does not show any output.

== CMD\_HELP\_UNCOUNCHANGED ==

Remarks:

This command is applied from the root of the workspace recursively.

Reading input from stdin:

The '^uncounchanged' command can read paths from stdin. To do this, pass a

single dash "-".

Example: cm ^uncounchanged -

Paths will be read until an empty line is entered.

This allows you to use pipe to specify for which unchanged files to undo

the checkout.

Example:

dir /S /B \*.c | cm ^uncounchanged -

(In Windows, undoes the checkout on all unchanged .c files in the

workspace.)

Examples:

cm ^uncounchanged . -^R

(Undoes checkouts of not changed files recursively on the current directory.)

cm ^unuc /home/myuser/mywk/project/templates -^R

(Undoes checkouts of not changed files recursively on the selected directory.)

== CMD\_DESCRIPTION\_UNDELETE ==

Undeletes an item using a specific revision.

== CMD\_USAGE\_UNDELETE ==

Usage:

cm ^undelete <revspec> <path>

revspec Specification of the revision whose contents will

be loaded in the workspace. (Use 'cm ^help ^objectspec' to

learn more about revision specs.)

path Restore path.

== CMD\_HELP\_UNDELETE ==

Remarks:

The item to undelete should not be already loaded in the workspace.

The '^undelete' operation is not supported for xlinks.

Example:

cm ^undelete ^revid:756 C:\mywks\src\foo.c

cm ^undelete ^itemid:68#^cs:2 C:\mywks\dir\myfile.pdf

cm ^undelete ^serverpath:/src#^br:/main C:\mywks\Dir

== CMD\_DESCRIPTION\_UNDOCHANGE ==

Undoes the changes on a path.

== CMD\_USAGE\_UNDOCHANGE ==

Usage:

cm ^undochange | ^unc <item\_path>[ ...] [-^R | -^r | --^recursive]

item\_path Items to apply the operation. Use a whitespace to separate

paths. Use double quotes (" ") to specify paths

containing spaces.

Use . to apply the operation to current directory.

Options:

-^R Applies the operation recursively.

== CMD\_HELP\_UNDOCHANGE ==

Remarks:

If an item is checked out or modified but not checked in and you do not

want to check it in, you can undo the changes using this command. The item

will be updated to the contents it had before.

Reading input from stdin:

The '^undochange' command can read paths from stdin. To do this, pass a

single dash "-".

Example: cm ^undochange -

Paths will be read until an empty line is entered.

This allows you to use pipe to specify for which files to undo changes.

Example:

dir /S /B \*.c | cm ^undochange -

(In Windows, undoes the changes of all .c files in the workspace.)

Examples:

cm ^unc .

(Undoes changes of the files on the current directory.)

cm ^undochange . -^R

(Undoes changes of the files recursively on the current directory.)

cm ^unc file1.txt "file 2.txt"

(Undoes changes of the selected files.)

cm ^unc c:\workspace\file.txt

(Undoes changes of the selected file.)

== CMD\_DESCRIPTION\_UNDO ==

Undoes changes in a workspace.

== CMD\_USAGE\_UNDO ==

Usage:

cm ^undo [<path>[ ...]] [--^symlink] [-^r | --^recursive] [<filter>[ ...]]

[--^silent | --^machinereadable [--^startlineseparator=<sep>]

[--^endlineseparator=<sep>] [--^fieldseparator=<sep>]]

path Path of the files or directories to apply the operation

to. Use double quotes (" ") to specify paths containing

spaces. Use a whitespace to separate paths.

If no path is specified, by default the undo

operation will take all of the files in the current

directory.

filter Applies the specified filter or filters to the given

paths. Use a whitespace to separate filters. See the

Filters section for more information.

Options:

--^symlink Applies the undo operation to the symlink and not

to the target.

-^r Executes the undo recursively.

--^silent Does not show any output.

--^machinereadable Outputs the result in an easy-to-parse format.

--^startlineseparator Used with the '--^machinereadable' flag, specifies

how the lines should start.

--^endlineseparator Used with the '--^machinereadable' flag, specifies

how the lines should end.

--^fieldseparator Used with the '--^machinereadable' flag, specifies

how the fields should be separated.

Filters:

If no flag is specified, by default all changes are undone, but the

paths can be filtered using one or more of the flags below.

If a file or directory matches one or more of the specified kinds of change,

all of the changes on said file or directory will be undone.

For example, if you specify both '--^checkedout' and '--^moved', if a file is

both checkedout and moved, both changes will be undone.

--^checkedout Selects checked-out files and directories.

--^unchanged Selects files whose content is unchanged.

--^changed Selects locally changed or checked-out files and

directories.

--^deleted Selects deleted files and directories.

--^moved Selects moved files and directories.

--^added Selects added files and directories.

== CMD\_HELP\_UNDO ==

Remarks:

The '^undo' command is dangerous - it undoes work in an irreversible way.

Once the ^undo has finished, it is not possible to recover the previous state

of the files and directories affected by it. If no path is specified

in the arguments, by default it will undo every change in the current

directory, but not recursively.

These are equivalent when executed from the /src directory:

/src

|- file.txt

|- code.cs

\- /test

|- test\_a.py

\- test\_b.py

cm ^undo

cm ^undo \*

cm ^undo file.txt code.cs /test

cm ^undo .

cm ^undo /src file.txt code.cs

If you want the operation to be recursive, you must specify the '-^r' flag.

To undo all of the changes below a directory (including changes affecting

the directory itself):

cm ^undo dirpath -^r

If dirpath is a workspace path, every change in the workspace will be

undone.

Deleted items:

To undo file and directory deletions, you must either specify the full path

of the item, or specify the containing directory and use the recursive ('-^r')

flag.

For example:

cm ^undo .

(Does NOT undo deletions in the current directory.)

cm ^undo . -^r

(Undoes all deletions (and other changes) in the current directory recursively.)

cm ^undo src/file.txt

(Undoes deletion (or other change) of src/file.txt.)

Examples:

cm ^undo . -^r

(Undoes all changes in the current directory recursively. If executed

from the workspace's root, undoes all changes in the entire workspace.)

cm ^co file.txt

cm ^undo file.txt

(Undoes the checkout on 'file.txt'.)

^echo ^content >> file.txt

cm ^undo file.txt

(Undoes the local change to 'file.txt'.)

cm ^undo src

(Undoes changes to the src directory and its files.)

cm ^undo src/\*

(Undoes changes in every file and directory contained in src, without

affecting src.)

cm ^undo \*.cs

(Undoes changes to every file or directory that matches \*.cs in the current

directory.)

cm ^undo \*.cs -^r

(Undoes changes on every file or directory that matches \*.cs in the current

directory and every directory below it.)

cm ^co file1.txt file2.txt

^echo ^content >> file1.txt

cm ^undo --^unchanged

(Undoes the checkout of unchanged 'file2.txt', ignoring locally changed

'file1.txt'.)

^echo ^content >> file1.txt

^echo ^content >> file2.txt

cm ^co file1.txt

cm ^undo --^checkedout

(Undoes the changes in checked-out file 'file1.txt', ignoring 'file2.txt' as

it is not checked-out.)

cm ^add file.txt

cm ^undo file.txt

(Undo the add of 'file.txt' making it once again a private file.)

^rm file1.txt

^echo ^content >> file2.txt

cm ^add file3.txt

cm ^undo --^deleted --^added \*

(Undoes the 'file1.txt' delete and 'file3.txt' add, ignoring the 'file2.txt'

change.)

== CMD\_DESCRIPTION\_LOCK\_UNLOCK ==

Undoes item locks on a lockserver.

== CMD\_USAGE\_LOCK\_UNLOCK ==

Usage:

cm ^lock ^unlock [<repserverspec>] <guid>[ ...]

repserverspec Repository server specification. (Use 'cm ^help ^objectspec'

to learn more about repserver specs.)

guid A list of item GUIDs to be unlocked. Use a whitespace to

separate GUIDs.

== CMD\_HELP\_LOCK\_UNLOCK ==

Remarks:

- The command uses the specified server to unlock the items.

- If no server is specified, the command tries to obtain a server from the

current workspace.

- If no server was calculated on the previous steps, the server is obtained

from the current Plastic SCM client configuration.

- Only the administrator of the server can run the 'cm ^unlock' command.

- To specify a GUID, the format should be the 32-digit separated by

hyphens format (optionally enclosed in braces):

{00000000-0000-0000-0000-000000000000}

or 00000000-0000-0000-0000-000000000000

Examples:

cm ^lock ^unlock 91961b14-3dfe-4062-8c4c-f33a81d201f5

(Undoes the selected item lock.)

cm ^lock ^unlock DIGITALIS:8084 2340b4fa-47aa-4d0e-bb00-0311af847865 \

bcb98a61-2f62-4309-9a26-e21a2685e075

(Undoes the selected item locks on lockserver named 'DIGITALIS'.)

cm ^lock ^unlock tardis@cloud 4740c4fa-56af-3dfe-de10-8711fa248635 \

71263c17-5eaf-5271-4d2c-a25f72e101d4

(Undoes the selected item locks on cloud lockserver named 'tardis'.)

== CMD\_DESCRIPTION\_UPDATE ==

Updates the workspace and downloads latest changes.

== CMD\_USAGE\_UPDATE ==

Usage:

cm ^update [<item\_path> | --^last]

[--^changeset=<csetspec>] [--^cloaked] [--^dontmerge] [--^forced]

[--^ignorechanges] [--^override] [--^recursewk] [--^skipchangedcheck]

[--^silent] [--^verbose] [--^xml[=<output\_file>]] [--^encoding=<name>]

[--^machinereadable [--^startlineseparator=<sep>]

[--^endlineseparator=<sep>] [--^fieldseparator=<sep>]]

[--^forcedetailedprogress]

item\_path Path to update.

Use . to apply update to current directory.

If no path is specified, then the current workspace is

fully updated.

--^last Changes the workspace selector from a changeset

configuration or a label configuration to a branch

configuration before updating.

The selector is changed to the branch the changeset or

label belongs to.

Options:

--^changeset Updates the workspace to a specific changeset.

(Use 'cm ^help ^objectspec' to learn more about

changeset specs.)

--^cloaked Includes the cloaked items in the update operation.

If this option is not specified, those items that are

cloaked will be ignored in the operation.

--^dontmerge In case an update merge is required during the update

operation, does not perform it.

--^forced Forces updating items to the revision specified in

the selector.

--^ignorechanges Ignores the pending changes warning message that is

shown if there are pending changes detected when

updating the workspace.

--^override Overrides changed files outside Plastic SCM control.

Their content will be overwritten with the server

content.

--^recursewk Updates all the workspaces found within the current

path. Useful to update all the workspaces contained

in a specific path.

--^skipchangedcheck The update checks if there are local changes in your

workspace before starting. If you always checkout

before modifying a file, you can use this check and

speed up the operation.

--^silent Does not show any output.

--^verbose Shows additional information.

--^xml Prints the output in XML format to the standard output.

It is possible to specify an output file.

--^encoding Used with the --^xml option, specifies the encoding to

use in the XML output, i.e.: utf-8.

See the MSDN documentation at

http://msdn.microsoft.com/en-us/library/system.text.encoding.aspx

to get the table of supported encodings and its format,

(at the end of the page, in the "Name" column).

--^machinereadable Outputs the result in an easy-to-parse format.

--^startlineseparator Used with the '--^machinereadable' flag,

specifies how the lines should start.

--^endlineseparator Used with the '--^machinereadable' flag,

specifies how the lines should end.

--^fieldseparator Used with the '--^machinereadable' flag,

specifies how the fields should be separated.

--^forcedetailedprogress Forces detailed progress even when standard output

is redirected.

== CMD\_HELP\_UPDATE ==

Remarks:

The '^update' command only downloads needed files.

The command assumes recursive operation.

When the '--^last' option is used, it is not necessary to specify a path.

In this case, the workspace the current working directory belongs to will

be updated.

(Remember that specifying this flag could cause the workspace

selector to be changed to a branch configuration if the selector was

previously pointing to a changeset or a label.)

Examples:

cm ^update

(Updates all in the current workspace.)

cm ^update .

(Updates current directory, and all children items.)

cm ^update . --^forced --^verbose

(Forces retrieval of all revisions.)

cm ^update --^last

cm ^update . --^machinereadable --^startlineseparator=">"

(Updates current directory and prints the result in a simplified

easier-to-parse format, starting the lines with the specified

strings.)

== CMD\_DESCRIPTION\_VERSION ==

Shows the current client version number.

== CMD\_USAGE\_VERSION ==

Usage:

cm ^version

== CMD\_HELP\_VERSION ==

== CMD\_DESCRIPTION\_WHOAMI ==

Shows the current Plastic SCM user.

== CMD\_USAGE\_WHOAMI ==

Usage:

cm ^whoami

== CMD\_HELP\_WHOAMI ==

== CMD\_USAGE\_WKTREENODESTATUS ==

Usage:

cm ^wktreenodestatus path1, path2, ...

== CMD\_DESCRIPTION\_WORKSPACE ==

Allows the user to manage workspaces.

== CMD\_USAGE\_WORKSPACE ==

Usage:

cm ^workspace | ^wk <command> [options]

Commands:

^list | ^ls

^create | ^mk

^delete | ^rm

^move | ^mv

^rename

To get more information about each command run:

cm ^workspace <command> --^usage

cm ^workspace <command> --^help

== CMD\_HELP\_WORKSPACE ==

Examples:

cm ^workspace ^create myWorkspace wk\_path

cm ^workspace ^list

cm ^workspace ^delete myWorkspace

== CMD\_DESCRIPTION\_WORKSPACE\_CREATE ==

Creates a new workspace.

== CMD\_USAGE\_WORKSPACE\_CREATE ==

Usage:

cm ^workspace | ^wk [^create | ^mk] <wk\_name> <wk\_path> [<rep\_spec>]

[--^selector[=<selector\_file>]

(Creates a new workspace.)

cm ^workspace | ^wk [^create | ^mk] <wk\_name> <wk\_path> --^dynamic --^tree=[<tree>]

(Creates a dynamic workspace. This feature is still experimental, and it's

only available for Windows.)

wk\_name The new workspace name.

wk\_path Path of the new workspace.

rep\_spec Creates the new workspace with the specified repository.

Repository specification: check 'cm ^help ^objectspec'.

Options:

--^selector Edits a selector for the new workspace.

If a selector file is specified, then sets the selector

for the new workspace from the specified file.

--^dynamic Creates a dynamic workspace. This feature is still

experimental, and it's only available for Windows.

Specifying this flag requires using the --^tree parameter.

--^tree Used with the '--^dynamic' flag, specifies the initial

point the dynamic workspace is going to load. It can

either be a branch, changeset, or label specification.

The workspace will later on use the repository in the

spec. (Use 'cm ^help ^objectspec' to learn more about specs.)

== CMD\_HELP\_WORKSPACE\_CREATE ==

Remarks:

- A workspace is a view of the repository mapped to the local filesystem.

The workspace selector defines the rules that specify workspace content.

Use 'cm ^showselector' to display a workspace selector or 'cm ^setselector'

to modify it.

- If neither rep\_spec nor '--^selector' is specified, then the workspace

will automatically be configured to use the first repository

(alphabetically) of the server configured in the client.conf file.

- The dynamic workspaces is a experimental feature (Windows only), and it

requires the plasticfs.exe program running.

Examples:

cm ^workspace ^create myworkspace c:\workspace

cm ^wk ^mk myworkspace /home/john/plastic\_view

(Creates 'myworkspace' workspace in Windows and in Linux respectively.)

cm ^wk mywktest c:\wks\wktest --^selector=myselector.txt

(Creates 'mywktest' workspace using the selector in 'myselector.txt' file.)

cm ^wk mywkprj c:\wks\wkprj myrep@^repserver:localhost:8084

(Creates 'mywkprj' workspace with the selected repository.)

cm ^wk mywkprj c:\dynwks\mywkprj --^dynamic --^tree=^br:/main@myrep@localhost:8084

(Creates dynamic 'mywkprj' workspace with the 'myrep@localhost:8084'

repository, pointing to '^br:/main' the first time it gets mounted.)

== CMD\_DESCRIPTION\_WORKSPACE\_DELETE ==

Deletes a workspace.

== CMD\_USAGE\_WORKSPACE\_DELETE ==

Usage:

cm ^workspace | ^wk ^delete | ^rm [<wk\_path> | <wkspec>] [--^keepmetadata]

wk\_path Path of the workspace to be deleted.

wkspec Specification of the workspace to delete. (Use

'cm ^help ^objectspec' to learn more about specs.)

Options:

--^keepmetadata Does not delete the metadata files in the .plastic

folder.

== CMD\_HELP\_WORKSPACE\_DELETE ==

Remarks:

This command deletes a workspace, specified by path or spec.

If no arguments are specified, current workspace will be assumed.

Examples:

cm ^workspace ^delete

(Removes current workspace.)

cm ^wk ^delete c:\workspace

cm ^workspace rm /home/danipen/wks

cm ^wk ^rm ^wk:MiWorkspace

cm ^wk ^rm ^wk:MiWorkspace@DIGITALIS

== CMD\_DESCRIPTION\_WORKSPACE\_LIST ==

Lists workspaces.

== CMD\_USAGE\_WORKSPACE\_LIST ==

Usage:

cm ^workspace | ^wk [^list | ^ls] [--^format=<str\_format>]

Options:

--^format Retrieves the output message in a specific format. See

Remarks for more info.

== CMD\_HELP\_WORKSPACE\_LIST ==

Remarks:

This command accepts a format string to show the output.

The output parameters of this command are the following:

{0} | {^wkname} Workspace name.

{1} | {^machine} Client machine name.

{2} | {^path} Workspace path.

{3} | {^wkid} Workspace unique identifier.

{4} | {^wkspec} Workspace specification using the format:

'wkname@machine'.

{^tab} Inserts a tab space.

{^newline} Inserts a new line.

Examples:

cm ^wk

(Lists all workspaces.)

cm ^workspace ^list --^format={0}#{3,40}

cm ^workspace ^list --^format={^wkname}#{^wkid,40}

(Lists all workspaces and shows the workspace name, a # symbol and the

workspace GUID field in 40 spaces, aligned to left.)

cm ^wk --^format="Workspace {0} in path {2}"

cm ^wk --^format="Workspace {^wkname} in path {^path}"

(Lists all workspaces and shows result as formatted strings.)

== CMD\_DESCRIPTION\_WORKSPACE\_MOVE ==

Moves a workspace.

== CMD\_USAGE\_WORKSPACE\_MOVE ==

Usage:

cm ^workspace | ^wk ^move | ^mv [<wkspec>] <new\_path>

Options:

wkspec Specification of the workspace to move. (Use

'cm ^help ^objectspec' to learn more about specs.)

new\_path Workspace will be moved to here.

== CMD\_HELP\_WORKSPACE\_MOVE ==

Remarks:

This command allows users to move a workspace to another location on disk.

Examples:

cm ^workspace ^move myWorkspace \new\workspaceDirectory

(Moves 'myWorkspace' to the specified location.)

cm ^wk ^mv c:\users\maria\wkspaces\newlocation

(Moves the current workspace to the new location.)

== CMD\_DESCRIPTION\_WORKSPACE\_RENAME ==

Renames a workspace.

== CMD\_USAGE\_WORKSPACE\_RENAME ==

Usage:

cm ^workspace | ^wk ^rename [<wk\_name>] <new\_name>

wk\_name Workspace to rename.

new\_name New name for the workspace.

== CMD\_HELP\_WORKSPACE\_RENAME ==

Remarks:

This command renames a workspace.

If no workspace name is supplied, the current workspace will be used.

Examples:

cm ^workspace ^rename mywk1 wk2

(Renames the workspace 'mywk1' to 'wk2'.)

cm ^wk ^rename newname

(Renames the current workspace to 'newname'.)

== CMD\_DESCRIPTION\_WORKSPACESTATUS ==

Shows changes in the workspace.

== CMD\_USAGE\_WORKSPACESTATUS ==

Usage:

cm ^status [<wk\_path>] [--^changelist[=<name>] | --^changelists] [--^cutignored]

[ --^header] [ --^noheader] [ --^nomergesinfo] [ --^head]

[--^short] [--^symlink] [ --^dirwithchanges] [--^xml[=<output\_file>]]

[--^encoding=<name>] [ --^wrp | --^wkrootrelativepaths]

[--^fullpaths | --^fp] [<legacy\_options>] [<search\_types>[ ...]]

[--^machinereadable [--^startlineseparator=sep]

[--^endlineseparator=sep] [--^fieldseparator=sep]]

Options:

wk\_path Path of the workspace where the search will be

performed.

--^changelist Shows changes in the selected changelist.

--^changelists Shows changes grouped in client changelists.

--^cutignored Skips the contents of ignored directories.

Requires the '--^ignored' search type. See the Search

types section for more information.

--^header Only prints the workspace status.

--^noheader Only prints the modified item search result.

--^nomergesinfo Doesn't print the merge info for changes.

--^head Prints the status of the last changeset on the branch.

--^short Lists only paths that contains changes.

--^symlink Applies the operation to the symlink and not to the

target.

--^dirwithchanges Shows directories that contain changes inside them

(added, moved, removed items inside).

--^xml Prints the output in XML format to the standard output.

It is possible to specify an output file.

--^encoding Used with the --^xml option, specifies the encoding to

use in the XML output, i.e.: utf-8.

See the MSDN documentation at

http://msdn.microsoft.com/en-us/library/system.text.encoding.aspx

to get the table of supported encodings and its format,

(at the end of the page, in the "Name" column).

--^wrp Print workspace root relative paths instead of

current directory relative paths.

--^fullpaths, --^fp Force printing absolute paths, overriding any other

path printing setting.

--^machinereadable Outputs the result in an easy-to-parse format.

--^startlineseparator Used with the '--^machinereadable' flag,

specifies how the lines should start.

--^endlineseparator Used with the '--^machinereadable' flag,

specifies how the lines should end.

--^fieldseparator Used with the '--^machinereadable' flag,

specifies how the fields should be separated.

Legacy options:

--^cset Prints the workspace status in the legacy format.

--^compact Prints the workspace status and changelists in the

legacy format.

--^noheaders When used in conjunction with the '--^compact' flag, the

changelist headers will not be printed. (Does not apply

to the new changelists format.)

Search types:

--^added Prints added items.

--^checkout Prints checkedout items.

--^changed Prints changed items.

--^copied Prints copied items.

--^replaced Prints replaced items.

--^deleted Prints deleted items.

--^localdeleted Prints locally deleted items.

--^moved Prints moved items.

--^localmoved Prints locally moved items.

--^percentofsimilarity=<value> Percent of similarity between two files in

order to consider them the same item. Used

in the locally moved search. Its default

value is 20%.

--^txtsameext Only those text files that have the same

extension will be taken into account by the

similarity content matching process during

the moved items search. By default, any

text file will be processed.

--^binanyext Any binary file will be taken into account

by the similarity content matching process

during the moved items search. By default,

only those binary files that have the same

extension will be processed.

--^private Prints non controlled items.

--^ignored Prints ignored items.

--^hiddenchanged Prints hidden changed items. (Includes

'--^changed')

--^cloaked Prints cloaked items.

--^controlledchanged This flag substitutes the following options:

'--^added', '--^checkout', '--^copied',

'--^replaced', '--^deleted', '--^moved'.

--^all This flag replaces the following parameters:

'--^controlledchanged', '--^changed',

'--^localdeleted', '--^localmoved', '--^private'.

== CMD\_HELP\_WORKSPACESTATUS ==

Remarks:

The '^status' command prints the loaded changeset on a workspace and gets

the changed elements inside the workspace.

This command can be used to show the pending changes in a workspace; the

type of changes that can be searched can be modified by using the command

parameters. By default, all changes are displayed, be they controlled

or local.

The percent of similarity parameter '--^percentofsimilarity' (-^p) is used

by the locally moved search to decide if two elements are the same item.

The default value is 20% but it can be adjusted.

It is possible to show workspace changes grouped by client changelists.

The '^default' changelist includes the changes that are not included in

other changelists. That being said, the changes the default changelist

will show depends on the search types flags specified.

Showing changes grouped by changelists requires showing controlled

changes too (items with status equal to '^added', '^checkout', '^copied',

'^replaced', '^deleted', or '^moved'). So, the '--^controlledchanged' option

will be automatically enabled when changelists are shown.

The default encoding for XML output is utf-8.

By default, this command will print current directory relative paths,

unless the '--^machinereadable' or '--^short' flags are specified. If

any of them are specified, the command will print absolute paths.

If '--^xml' flag is specified, workspace root relative paths will

be printed (unless the '--^fp' flag is also specified, printing

absolute paths instead).

Examples:

cm ^status

(Prints the working changeset and also all item types changed in the

workspace, except the ignored ones.)

cm ^status --^controlledchanged

(Prints the working changeset and also the items that are checkedout, added,

copied, replaced, deleted, and moved.)

cm ^status --^added

(Prints only the working changeset and the added items inside the workspace.)

cm ^status c:\workspaceLocation\code\client --^added

(Prints the working changeset and the added items under the specified path

recursively.)

cm ^status --^changelists

cm ^status --^changelist

(Shows all the workspace changes grouped by client changelists.)

cm ^status --^changelist=pending\_to\_review

(Shows the changes on the changelist named 'pending\_to\_review'.)

cm ^status --^changelist=default --^private

(Shows the changes in the 'default' changelist, showing private items, along

with items with controlled changes, if any.)

cm ^status --^short --^changelist=pending\_to\_review | cm ^checkin -

(Checkins the changes in the changelist 'pending\_to\_review'.)

cm ^status C:\workspaceLocation --^xml=output.xml

(Gets the status information in XML format and using utf-8 in the file

output.xml.)

cm ^status --^ignored

(Shows all ignored items.)

Output:

/main@myrepo@local (^cs:2 - ^head)

^Added

Status Size Last Modified Path

^Ignored 0 bytes 19 seconds ago out\app.exe

^Ignored 48 seconds ago src

^Ignored 0 bytes 48 seconds ago src\version.c

cm ^status --^ignored --^cutignored

(Shows ignored files whose parent directory is not ignored and ignored

directories but not their contents.)

Output:

/main@myrepo@local (^cs:2 - ^head)

^Added

Status Size Last Modified Path

^Ignored 0 bytes 19 seconds ago out\app.exe

^Ignored 48 seconds ago src

== CMD\_DESCRIPTION\_XLINK ==

Creates, edits, or displays details of an Xlink.

== CMD\_USAGE\_XLINK ==

Usage:

cm ^xlink [-^w] [-^rs] <xlink\_path> / (<csetspec> | <lbspec> | <brspec)>

[<expansion\_rules>[ ...]]

(Creates an Xlink.)

cm ^xlink [-^rs] <xlink\_path> /<relative\_path> (<csetspec> | <lbspec> | <brspec>)

[<expansion\_rules>[ ...]]

(Creates a readonly partial Xlink pointing to /<relative\_path> instead of

the default root / .)

cm ^xlink -^e <xlink\_path> (<csetspec> | <lbspec> | <brspec>)

(Edits an Xlink to change the target specification.)

cm ^xlink -^s|--^show <xlink\_path>

(Shows the Xlink information including the expansion rules.)

cm ^xlink -^ar|--^addrules <xlink\_path> <expansion\_rules>[ ...]

(Adds the given expansion rules to the Xlink).

cm ^xlink -^dr|--^deleterules <xlink\_path> <expansion\_rules>[ ...]

(Removes the given expansion rules from the Xlink).

xlink\_path This is the directory in the current workspace where

the linked repository will be mounted (when creating an

Xlink) or is mounted (when editing an Xlink).

csetspec The full target changeset specification in the remote

repository.

This determines what version and branch is loaded in the

workspace for the linked repository.

(Use 'cm ^help ^objectspec' to learn more about changeset

specs.)

lbspec The full label specification in the remote repository.

(Use 'cm ^help ^objectspec' to learn more about label

specs.)

brspec The full branch specification in the remote repository.

This uses the current changeset where the specified

branch is pointing to. (Use 'cm ^help ^objectspec' to

learn more about branch specs.)

-^e Edits an existing Xlink to change the target changeset

specification.

-^s | --^show Shows information about the selected Xlink.

-^ar | --^addrules Adds one or more expansion rules to the selected Xlink.

-^dr | --^deleterules Deletes one or more expansion rules from the selected

Xlink.

expansion\_rules To specify one or more expansion rule. Each expansion

rule is a pair branch-target branch:

^br:/main/fix-^br:/main/develop/fix

Options:

-^w Indicates that the Xlink is writable. This means that

the contents of the target repository can be modified

through branch autoexpansion.

-^rs Relative server. This option allows creating a relative

Xlink that is independent of the repository server. This

way, Xlinks created in replicated repositories in

different servers will be automatically identified.

== CMD\_HELP\_XLINK ==

Remarks:

This command creates an Xlink to a given changeset. By default, a read-only

Xlink is created. This means that the contents loaded in the workspace

inside the Xlink cannot be modified. To be able to make changes in the

Xlinked content, create a writable Xlink instead (using the '-^w' option).

It is possible to use a simplified syntax of the command when editing the

target changeset of an Xlink. This way, the only required parameter is the

new target changeset. The rest of parameters of the Xlink will not be

modified.

Branch auto-expansion:

When a change is made in any writable-xlinked repositories ('-^w' option), a

new branch needs to be created in the target repository. The name of the

new branch is based on the checkout branch defined in the top-level

repository. To determine the name of the branch to use, these rules apply:

1) A check is made to see if a branch with the same full name exists

in the target repository:

- If it exists, this is used as the checkout branch.

- If it does not exist, the branch name is built this way:

- Name of the branch of the target Xlinked changeset + short name of

the checkout branch (last part).

- If this branch exists, it is used as the checkout branch.

- Otherwise, the branch is created and the branch base is set to the

Xlinked changeset.

2) A new version of the Xlink is created in the branch on the parent

repository pointing to the new changeset in the Xlinked repository.

Finally, the complete Xlink structure is kept up to date with the latest

changes in the right versions.

Examples:

cm ^xlink code\firstrepo / 1@first@localhost:8084

(Creates an Xlink in folder 'firstrepo' in the current workspace where the

changeset '1' in the repository 'first' will be mounted.)

cm ^xlink opengl\include /includes/opengl 1627@includes@localhost:8087

(Creates a readonly partial Xlink in directory 'opengl\include' in the

current workspace where the path '/includes/opengl' in changeset '1627' in

the repository 'includes' will be mounted as root. It means that whatever

is inside '/includes/opengl' will be mounted in 'opengl\include' while the

rest of the repository will be ignored.)

cm ^xlink -^w -^rs code\secondrepo / ^lb:LB001@second@localhost:8084

(Creates a writable and relative Xlink in folder 'secondrepo' in the

current workspace where the label 'LB001' in the repository 'second' will

be mounted.)

cm ^xlink code\thirdrepo / 3@third@localhost:8087 ^br:/main-^br:/main/scm003

(Creates an Xlink in folder 'thirdrepo' in the current workspace where the

changeset '3' in the repository 'third' will be mounted.)

cm ^xlink -^e code\secondrepo ^br:/main/task1234@second@localhost:8084

(Edits the Xlink 'code\secondrepo' to change the target repository by

linking the branch 'main/task1234' in the repository 'second'.)

cm ^xlink --^show code\thirdrepo

(Shows information of the Xlink 'code\thirdrepo' including its expansion

rules if exist).

cm ^xlink -^ar code\secondrepo ^br:/main-^br:/main/develop ^br:/main/fix-^br:/main/develop/fix

(Adds two expansion rules to the xlink 'code\secondrepo'.)

cm ^xlink -^dr code\secondrepo ^br:/main/fix-^br:/main/develop/fix

(Deletes the expansion rule from the xlink 'code\secondrepo').

== CMD\_USAGE\_AUTOCOMPLETE ==

Usage:

cm ^autocomplete ^install

(Installs 'cm' command completion in the shell.)

cm ^autocomplete ^uninstall

(Uninstalls 'cm' command completion from the shell.)

cm ^autocomplete --^line <shell\_line> --^position <cursor\_position>

(Returns autocomplete suggestions for 'shell\_line' to be inserted at

'cursor\_position'. This command is not intended to be used by the final

user, but it is documented in case you want to extend autocompletion

support for your shell of choice.)

shell\_line The line the user has written into the shell when the

autocompletion was requested.

In Bash, it is at the COMP\_LINE environment variable.

In PowerShell, it is at the $wordToComplete variable.

cursor\_position The position of the cursor when the autocompletion was

requested.

In Bash, it is at the COMP\_POINT environment variable.

In PowerShell, it is at the $cursorPosition variable.