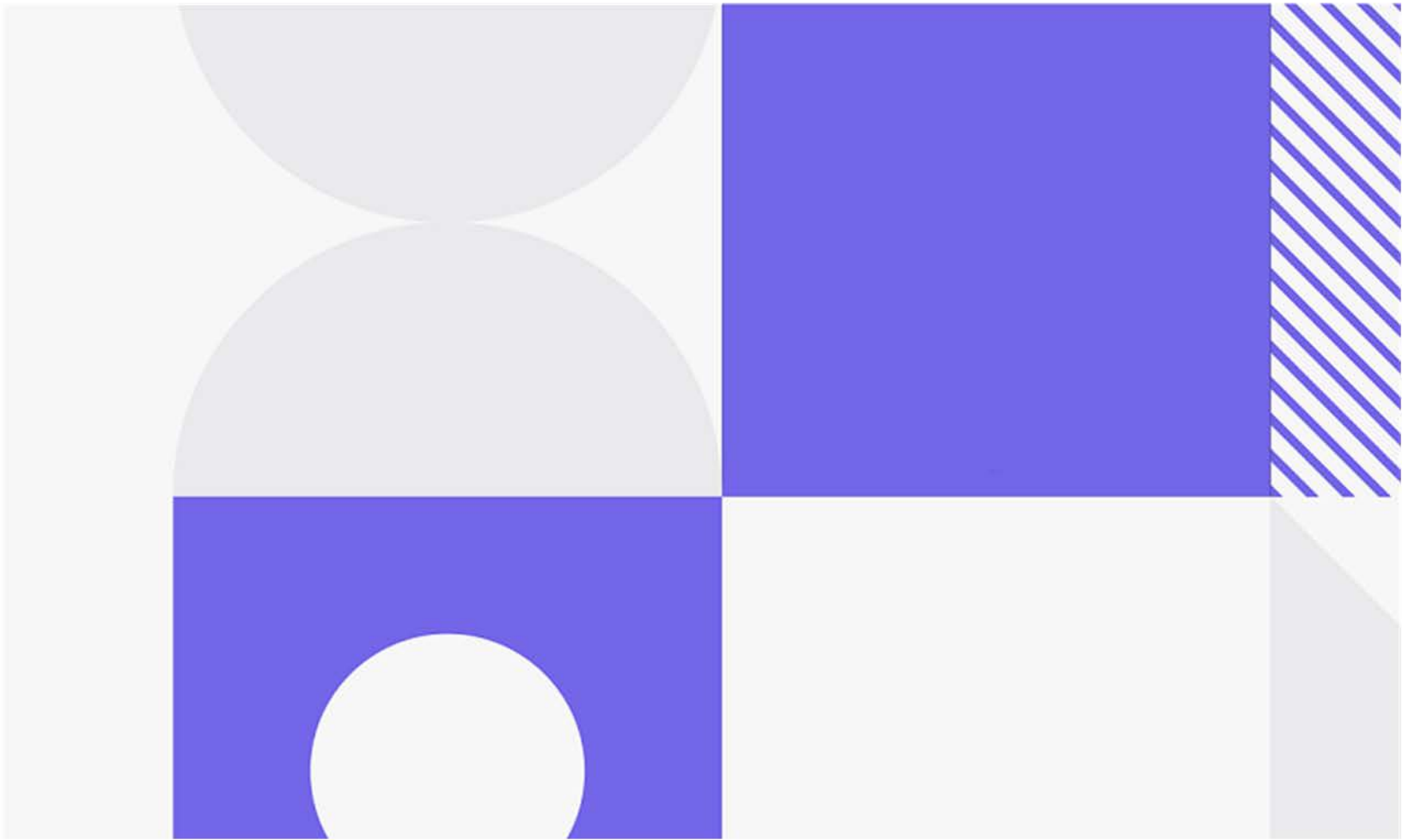
opentext™

PVCS Version Manager

Software version: 25.4

Command-Line Reference Guide



Product version: 25.4

Last updated: December 4, 2025

The most recent edition of this manual (with errata included) can be downloaded here:
https://www.microfocus.com/documentation/pvcs-version-manager/25.4/VMCLIRG.pdf

Table of Contents

Chapter 1
Commands

Using Commands

Use Version Manager commands

Version Manager consists of a number of utilities, which you execute by issuing *commands*. Command-line users may perform operations on archives using these commands instead of using the desktop client.

NOTE If you use symbolic links to specify paths in Version Manager commands, the command takes longer to complete. To speed up the process, specify full paths rather than symbolic links, or add the fully qualified paths of Version Manager commands to your environment.

This section contains the following topics about issuing commands:

Command Types

The commands are grouped into four categories: primary, secondary, security-related, and auxiliary.

Use this command...	To do this...
`get`	Check a revision out of an archive.
`put`	Check a workfile into an archive.
`vcs`	Change archive information.
`vdiff`	Show the differences between two revisions or workfiles.
`vlog`	Generate reports on archive information.

Use this command...	To do this...
`ident`	Display keywords in a file.
`regen`	Regenerate a file from a delta file.
`vcompres`	Compress or uncompress archive records.
`vdel`	Delete a revision.
`vjournal`	Display a journal report.
`vmrg`	Merge changes in two revisions.

Use this command...	To do this...
vpromote	Move a revision up a level in the promotion hierarchy.
vsplit	Split archives into separate revision and metadata stores for use with the revision library feature of the Version Manager File Server.
vsql	Write command files for exporting archive data into SQL database systems.
vtransfer	(For archives mapped to a Version Manager File Server only.) Export archives to a zip file, import archives from a zip file, rename archives, move archives, fix the revision library path for archives.

Use this command...	To do this...
makedb	Create an access control database.
readdb	View the contents of an access control database.
vconfig	Embed configuration information in the VCONFIG file.

Use this command...	To do this...
filt	Filter text files regardless of programming language (Windows only).
printenv	Display the values of environment variables (Windows only).
rse	Redirect a command's error messages (Windows only).

NOTE Use `command -h` to display a list of the options available for a command.

Syntax

command [-*option*[*parameter*...]...] *file_name*...

Special Consideration

Spaces are not allowed between an option and a parameter.

Related Topics

For information about...	See...
How Version Manager searches for archives	"How Version Manager Searches for Archives" on page 15
How Version Manager uses suffixes to determine file names	"Suffix Translation" on page 123
How to specify archive names with wildcards	"FirstMatch directive" on page 151
How to escape characters in specifications	"Escape Characters in Specifications" on page 16

Specifying File Names

Specify archives and workfiles

When you use Version Manager commands, you specify the names of the files on which the commands operate. When using commands, you can specify the following:

- The workfile name. Version Manager automatically determines the name of the corresponding archive based on the settings in the configuration file.
- The archive name. In most cases, you need to specify the archive name; and are *required* to specify the archive name in some instances.
- Both the workfile and archive names.

Each of these options is discussed in the following sections in more detail. For more information on the algorithm Version Manager uses when searching for workfiles or archives, see "How Version Manager Searches for Archives" on page 15.

In order to execute a command successfully, you must specify the files using the appropriate syntax:

Syntax

```
[[ar_path][\ar_name.ext][([wrk_path]wrk_name.ext)]
```

Or

```
[wrk_path]wrk_name.ext]
```

To use the command-line interface on a file name containing special characters (such as left or right parenthesis characters), make sure that the separator characters defined to separate the archive and workfile name or path name are *not* characters used in a file name. See "Specifying Both Workfiles and Archives" on page 11.

Specifying Revisions and Workfiles

There are many ways to specify revisions and workfiles from the command line. You can specify a file name or a fully qualified path name as a parameter. The following are examples of file specification from the command line.

Examples

- To check out a revision with lock into the current directory.

UNIX and DOS

```
get -l test.c
```

- To check in a workfile from the current directory.

UNIX and DOS

```
put test.c
```

- To check in a workfile from the specified directory.

DOS

```
put D:\dir\dir\test.c
```

UNIX

```
put /usr/testdir/test.c
```

- To check out a revision to the specified directory.

DOS

```
get F:\dir\dir\test.c
```

UNIX

```
get /usr/testdir/test.c
```

- To check in all workfiles with a .c file extension from the current working directory.

DOS and UNIX

```
put *.c
```

- To check in all workfiles with a .cbl extension from the specified directory.

DOS
```
put F:\directory\*.CBL
```

UNIX
```
put /usr/testdir/*.cbl
```

Specifying Archives

You must specify the archive name when performing an operation, such as in the following cases.

- To perform a wildcard operation across all archives specified by the VCSDir directive.

DOS
```
get *.c-arc
```

UNIX
```
get \*.c-arc
```

- When specifying a workfile that corresponds to the archive that you wish to operate on does not exist in the current working directory.

DOS
```
get test.c-arc(D:\Workdir)
```

UNIX
```
get test.c-arc\(/usr/workdir\)
```

See "Specifying Both Workfiles and Archives" on page 11 for more information on specifying files with special characters.

- When applying a version label to all the files in your system.

DOS
```
vcs -vlatest *-arc
```

UNIX
```
vcs -vlatest \*-arc
```

- When checking out files to an empty directory.

DOS
```
get *-arc
```

UNIX
```
get \*-arc
```

- When updating the access list of a group of files in a certain directory.

DOS
```
vcs -adev,qa *-arc
```

UNIX
```
vcs -adev,qa \*-arc
```

Also, if you need to use wildcards when specifying archive names, you must remember to specify the archive extension instead of the workfile extension. For example:

Use this syntax...	To do this...
`get *.c_v`	Check out revisions from all archives with .C_V extensions into the current directory.
`get J:\VCS\SUB1\*.asv`	Check out revisions from all archives with .ASV extensions from the specified directory into the current directory.

For more information on using wildcards when specifying file or archive names, see "Specifying Names Using Wildcards" on page 13.

Specifying Both Workfiles and Archives

To specify both the archive and the workfile, use the following syntax:
```
get -l test.c_v(test.c)
```

Version Manager considers the file name enclosed in parentheses, such as (`main.c`), as the workfile, and any name or path name to the left of a file in parentheses, such as `main.c_v(main.c)`, as the archive. Version Manager uses this convention for workfiles and archives, regardless of the path or extension.

To use the command-line interface on a file name containing special characters (such as left or right parenthesis characters), make sure that the character defined to separate the archive and workfile are *not* characters used in a file name.

Change the separator character by setting two environment variables:

- `PVCS_LEFT_SEPARATOR`
- `PVCS_RIGHT_SEPARATOR`

The left and right separator characters can be two different characters (such as [and]), or the same character (such as !).

An example of setting the environmental variables to the left and right bracket using a UNIX C shell would be as follows:

```
setenv PVCS_LEFT_SEPARATOR [
setenv PVCS_RIGHT_SEPARATOR ]
```

The following example uses the new separator characters in an archive and workfile specification, where the workfile is being assigned a different name.

```
get test(1).c-arc[testfile.c]
```

The following example is an archive and workfile specification after the separator characters have both been changed to ! and where the workfile is being assigned different name.

```
get test(1).c-arc!testfile.c!
```

Note that your operating system or shell may require you to escape the character you specify as the separator with backslashes (\) or quotation marks (").

```
get "test(1).c-arc\!testfile.c\!"
```

File Specification Examples

Specify both the archive and the workfile names in the following situations:

To check out the C files into the specified directory

```
 DOS     get *.C_V(C:\Work)
UNIX     get \*.c_v \(/usr/work\)
```

- To specify where PUT should create archives

```
 DOS     put F:\Dir\test.c-arc(I:\Workdir\test.c)
UNIX     put /usr/dir/test.c-arc\(/usr/workdir/test.c\)
```

- To assign workfiles a different file name

```
 DOS     get main.c_v(trial.c)
UNIX     get main.c_v\(trial.c\)
```

- To specify a location for the workfiles other than the current directory

DOS
```
get test.c-arc(D:\Workdir)
```

UNIX
```
get test.c-arc\(/usr/workdir\)
```

- To create archives in the specified directory for all .C files in the current directory

DOS
```
put F:\Archives(*.c)
```

UNIX
```
put /usr/archives\(*.c\)
```

To specify a path for the workfiles only, add the IgnorePath directive to your MASTER.CFG or VCS.CFG configuration file. This prevents Version Manager from trying to find the archives in the workfile directory.

To limit the search for archives to the first VCSDIR found with a wildcard character match, set the FirstMatch directive.

Special Considerations

- To use the command-line interface on a file name containing special characters (such as left or right parenthesis characters), make sure that the character defined to separate the archive and workfile are *not* characters used in a file name. See "Specifying Both Workfiles and Archives" on page 11 for more information.
- Specifications can contain spaces, as long as they are enclosed in quotation marks. For example:
```
get -l "a space.c_v(a space.c)"
```
- **For UNIX users**: Your operating system or shell may require you to escape the parentheses in file specifications with backslashes (\).

Related Topics

For information about...	See...
How Version Manager searches for archives	"How Version Manager Searches for Archives" on page 15
How Version Manager uses suffixes to determine file names	"Suffix Translation" on page 123
Specify a path for workfiles only	"IgnorePath directive" on page 153
Specify archive names with wildcards	"FirstMatch directive" on page 151
Escaping special characters	"Escape Characters in Specifications" on page 16

Specifying Names Using Wildcards

A *wildcard* is a character that represents one or more characters. Use wildcards to execute a command on multiple files. You can use wildcards to specify files for most Version Manager commands.

Version Manager makes the following matches when you use the wildcard characters * and ?:

Character	Matches
?	Any single character
*	Zero or more characters in a file name or extension

Using wildcards in commands

When you specify file names using wildcards, you should first determine whether to specify the workfile or the archive name. In most instances, if you want to operate on archives that correspond to workfiles in the current directory, use a wildcard specification that matches the workfiles. To operate on archives located in one of the VCSDir directories, use a wildcard specification that matches the archive suffix.

Version Manager commands can compute an archive name from a workfile name, and vice versa. For this reason, you should be careful when using wildcards. For example, if you enter this command:

```
get *.c
```

The *.C expands to a list of all .C file names in the current working directory. Version Manager then searches archive directories specified by the VCSDir directive to find the corresponding archive for each file in the current working directory. If you intended to retrieve all *workfiles* in the archive repository (specified by VCSDir) whose names contain the .C extension, use the following command instead:

```
get *.c_v
```

To retrieve a copy of all files stored in archives, use the command:

```
get *.??v
```

Patterns that match ArchiveSuffix

When a Version Manager command encounters a wildcard specification, it examines the extension to see if it might be an archive extension (based on the value currently specified for ArchiveSuffix). If it is, Version Manager assumes you want to perform the command on all archives that match the wildcard pattern.

If the wildcard pattern has a path element, the command operates on archives in that directory. If it does not contain a path element, the command operates on each matching archive in each of the VCSDir directories, if any.

If the extension does not match the current ArchiveSuffix, the command operates only on files in the specified directory or in the current directory if there is no path element.

Special Considerations

- Spaces are not allowed between the separator character and the archive or workfile specification.
- **For UNIX users**: Enclose wildcard file specifications in single or double quotation marks (' or "), or use a backslash (\).

Related Topics

For information about...	See...
Specifying files and archives	"Specifying File Names" on page 10
Where the program looks for archives	"VCSDir directive" on page 175
How the program searches for files	"How Version Manager Searches for Archives" on page 15

For information about...	See...
Using escape characters in specifications	"Escape Characters in Specifications" on page 16
How Version Manager matches archive extensions	"ArchiveSuffix directive" on page 122
Changing the file search order	"FirstMatch directive" on page 151
Using special characters on the UNIX command line	"How Version Manager Searches for Archives" on page 15

How Version Manager Searches for Archives

How Version Manager finds files

If you do not specify files using the proper syntax identified in "Specifying File Names," on page 10, Version Manager uses the algorithm below to search for the file specified in the command. Version Manager uses this search algorithm if your file specification is ambiguous (i.e., you specify the workfile but not the archive, or you don't specify a path).

Figure 1-1. Archive Searching Algorithm

IF Only the workfile name is specified
THEN Derive the archive name using current suffix template
- **IF** The workfile name contains a path
- **THEN** Check that directory for the archive
  - **IF** The archive is found
  - **THEN** Use it
  - **ELSE** Display error message
- **ELSE** The workfile name contains no path
  Check the VCSDir directories
  Check the current directory
  - **IF** The archive is found
  - **THEN** Use it
  - **ELSE** Display error message

- **IF** The workfile name contains a path and the IGNOREPATH directive is set
- **THEN** Check that directory for the archive
  - **IF** The archive is found
  - **THEN** Use it
  - **ELSE** Display error message

- **ELSE IF** Check the VCSDir directories
  Check the current directory
  - **IF** The archive is found
  - **THEN** Use it
  - **ELSE** Display error message

ELSE IF Only the archive name is specified

THEN
- **IF** The archive name contains a path
- **THEN** Check that directory for the archive
  - **IF** The archive is found
  - **THEN** Use it
  - **ELSE** Check workfile

ELSE The archive name does not contain a path
Check the VCSDir directories
Check the current directory
IF The archive is found
THEN Use it
ELSE Display error message

If you supply both the archive and workfile names in a command, the program uses the same rules to locate the archive as it does when you only supply the archive name. When the program finds the workfile, however, it gives the checked out revision the workfile name you specified rather than the name stored in the archive.

Workfile searching

If a command requires a workfile name and you specify the archive name, it reads the workfile name from the archive and looks for it in the current directory.

Where archives are created

When you first check in a workfile, Version Manager creates the archive in a directory determined by the rules above. It strips any directory reference from the workfile name before storing it in the archive.

Special Consideration

The IgnorePath directive causes Version Manager to search the directories specified by the VCSDir directive when you specify an explicit workfile name, even if you specify a path name for the workfile.

Related Topics

For information about...	See...
Directives that affect the file searching algorithm	"IgnorePath directive" on page 153 "VCSDir directive" on page 175
Specifying workfile and archive names	"Specifying File Names" on page 10

Escape Characters in Specifications

Treat special characters literally

To *escape* a character is to direct Version Manager to treat a special character literally. Occasionally you need to escape characters when entering commands, or in Version Manager files (such as list files). Escape characters are not specific to Version Manager; they are a requirement of the operating system's shell.

Typically, you escape a character by preceding it with a backslash (\) or by enclosing it in quotation marks (also called *quoting*).

Some characters, called *control characters*, have a special meaning when used on the command line, in configuration files, and in list files. Control characters include double quotation marks ("), single quotation marks ('), backslashes (\), and pound signs (#).

Escape Characters in UNIX Commands

Quote or escape special characters

Be careful to quote or use escape characters in commands that include workfile names, wildcards, floating version labels, or date/time and revision ranges.

These features use characters such as asterisks (*), question marks (?), and parentheses ( ), which have a special meaning to UNIX shells.

For example, the following commands:

```
get *.??v
vcs -vunix4.0:* quux.c_v
```

```
get gemsup.c_v(build/gemsup.c)
vjournal -ustand -d8/15/91*8/22/91 contract.c
```

Should be changed as follows for UNIX systems:

```
get "*.??v"
vcs -vunix4.0:\* quux.c_v
get gemsup.c_v\(build/gemsup.c\)
vjournal -ustand -d8/15/91\*8/22/91 contract.c
```

Use shell rules for escaping characters

Version Manager relies on the UNIX shell to process quoted or escaped arguments. Different shells have different sets of special characters that must be escaped under the UNIX version of Version Manager. When in doubt, follow the rules for the shell you're using.

Related Topics

For information about...	See...
Where to use escape characters	"Configuration Files" on page 111 "Specifying File Names" on page 10 "List and Group Files" on page 21 "NewLine directive" on page 161 "PathSeparator directive" on page 163 "Specifying Version Labels in Commands" on page 17

Specifying Version Labels in Commands

Identify a revision symbolically

A *version label* is a symbolic name assigned to a revision in one or more archives. Version labels provide a convenient way to refer to different revisions by a single name.

Use version labels instead of revision numbers

You can use version labels in commands anywhere you would specify a revision number. For example, you can use version labels with the GET, VDEL, VLOG, and VMRG commands.

A version label typically identifies a new version of an entire software system (such as a major build). Each revision can have multiple version labels associated with it, but a specific version label can only be associated with one revision of an archive.

The following rules apply to version labels:

- Version labels are case-sensitive.
- Version labels can contain any printable characters except colons (:).
- If a version label begins with a digit, do one of the following:
  - Precede the version label with a backslash (\).
  - Use the `-v` option to specify the version label instead of the `-r` option. For example, the following commands are equivalent:

DOS

```
get -r\1.1a
get -v1.1a
```

UNIX

```
get -r\\1.1a
```

NOTE It is best practice to use version labels that do **NOT** look like revision numbers.

- If a version label contains a space or tab, enclose the version label in double quotation marks (""), as shown below:

  ```
  get -v"first release" test.c
  ```

Specifying Revision Ranges Using Version Labels

Operate on all revisions between two labels

When you use the following commands, you can specify a range of revisions by using version labels:

```
vdel -v
vdel -r
vlog -v
vlog -br
```

To learn how to specify a range of revisions using version labels, see "Specifying Revision Ranges" on page 19.

Special Consideration

Avoid using a plus (+) or minus (-) sign in version labels; otherwise, you will get unexpected results when you specify a relative revision.

Related Topics

For information about...	See...
Assigning version labels	"PUT command" on page 38 "VCS command" on page 56
Specifying revision numbers or a range of revisions using version labels	"Specifying Revisions in Commands" on page 18
Using version labels when merging revisions	"VMRG command" on page 81
Related privileges	"ModifyVersion privilege" on page 201

Specifying Revisions in Commands

Maintain history of workfile changes

A *revision* is a particular iteration of a workfile. Each time you modify a workfile and check it back into an archive, Version Manager creates a new revision and assigns it a new revision number. You can specify the default revision, a range of revisions, relative revisions, or first and last revisions using Version Manager commands.

The latest revision in a line of development is called the tip revision. Both the trunk (the main line of development) and each branch have their own tip revisions. If you lock a revision other than the tip revision, Version Manager creates a branch when you check it back in.

Selecting Revisions by Default

Select a revision other than the trunk tip

By default, when you don't explicitly specify a particular revision or version label with a command, Version Manager operates on the latest revision on the trunk.

You can use the DefaultVersion directive to control which version Version Manager operates on by specifying a version label that is assigned to a particular revision. That revision becomes the default rather than trunk's tip revision. This directive is especially useful when you want to make a branch revision the default rather than the trunk tip.

You can override the DefaultVersion directive on the command line by using a hyphen (-) after any command that accepts the revision number or a version label option.

Example

```
get -r- text.c
```

Specifying Revision Ranges

Select consecutive revisions

The following commands accept a revision range as a parameter:

vdel -r
vdel -v
vlog -br
vlog -r

You can specify an inclusive range of revision numbers to specify a range of revisions on which to operate. Use the following rules when specifying revision ranges:

Specify this...	To select...
*revision	All revisions on the same branch up to and including the specified revision.
revision*	This revision and all following revisions on this branch.
revision*revision	All revisions on this branch within the inclusive range.
branch	All revisions on this branch.
*	All trunk revisions.

NOTE Add a plus sign (+) to the revision range to include *all* revisions on *all* branches beginning from the revisions in the specified range. You may also specify version labels in combination with revision numbers to specify a range of revisions.

Examples

The following examples illustrate how to use version labels to specify a range for the VDEL command. You could use the same command-line syntax with the VLOG command to customize archive reports.

- To delete all revisions between and including the version label `alpha` and the version label `beta`:

DOS

```
vdel -valpha*beta prog.c
```

UNIX

```
vdel -valpha\*beta prog.c
```

- To delete all revisions between revision 1.2 and the version label `beta` in `prog.c`:

DOS

```
vdel -r1.2*beta prog.c
```

UNIX

```
vdel -r1.2\*beta prog.c
```

- To delete all revisions between and including the version label `alpha` and the version label `beta` in `prog.c`, including any branches that begin from those revisions in `prog.c`:

DOS
```
vdel -valpha*beta+ prog.c
```

UNIX
```
vdel -valpha\*beta+ prog.c
```

Specifying a Relative Revision

Specify a revision relative to a revision or version label

To specify one or more revisions relative to a named revision or version label, use the following syntax:

[*revision* | *version*]+ | *-value*

The *value* parameter can be any number from 0 through 65535.

Examples

- To check out the latest revision of ERROR.C prior to revision 3.0:
  `get -r3.0-1 error.c`
- To delete the second revision beyond the version label testdev:

  vdel -vtestdev+2
- To generate a report on the revision preceding version label testdev:

  vlog -bvtestdev-1

Specifying the First and Last Revisions

To reference the first (+) or last (-) revision on a branch or, use a plus or minus sign and the number sign (#):

[*revision* | *version*]+ | *-#*

Examples

- To check out the latest revision of test.c with the major number 2:
  `get -r2 test.c`
- To check out the latest revision on branch 1.3.1:
  `get -r1.3.1 test.c`
- To delete the last revision on a branch associated with the version label `testdev`:
  `vdel -vtestdev-#`
- To display revision information for each workfile with the extension .doc between revision 3.2 and 3.9, excluding those revisions:

DOS
```
vlog -br -r3.2+1*3.9-1 *.doc
```

UNIX
```
vlog -br -r3.2+1\*3.9-1 \*.doc
```

- In the following examples, `testdev` is a version label in test.c. To lock `testdev`, you would type:
  `vcs -ltestdev test.c`

  To lock the revision after `testdev`, you would type:
  `vcs -ltestdev+1 test.c`
- To lock the first revision on the same branch that `testdev` is on:
  `vcs -ltestdev+# test.c`
- To delete every revision, beginning with the first and ending with the revision prior to version `X`:

```
vdel -rx+#*x-1 test.c
```

- To display the differences between the revision with version label X and the tip revision on the same branch as X:
  ```
  vdiff -rx -rx-# test.c
  ```

Related Topics

For information about...	See...
Using Version Manager commands	"Using Commands" on page 8
Specifying a default version	"DefaultVersion directive" on page 135
Specifying a new revision number	"PUT command" on page 38

List and Group Files

Specify additional command-line options in a file

A *list file* contains additional command-line options for Version Manager commands. Use the *@list_file* command-line option to direct Version Manager to read a list file before scanning the rest of the command line.

You can create list files to ease users from having to remember Version Manager command-line options and to standardize the options used.

Group files are a special type of list file that Version Manager finds automatically. You can keep all group files in one directory for easier maintenance.

List file format

List files can contain any command-line parameters. Lines in a list file can be of any length. Use white space to separate elements in the file. To continue lines onto succeeding lines, use a backslash (\) as the last character. Add a comment to a list file using the following syntax:
-#"*comment*"

List files can contain additional list files. They can be nested to approximately 20 levels.

Using List Files

Simplify operations on multiple files

You can use list files to perform an operation on multiple files in several directories. For example, you could simplify the task of assigning a version label to the tip revisions of all archives in a project by creating a list file containing the explicit path names of all archives in the project.

For example, you could create the following list file, named LOCALFIL.LST:

DOS

```
k:\math\include\i_o.c_v
k:\math\include\mathhdr.c_v
k:\math\include\mathlib.c_v
k:\math\include\mathsys.c_v
```

UNIX

```
/math/include/i_o.c_v
/math/include/mathhdr.c_v
/math/include/mathlib.c_v
/math/include/mathsys.c_v
```

You could use this list file to perform commands similar to the following:

```
vcs -v"NEWREV" @localfil.lst
```

NOTE A shortcut for the above example would be to set the VCSDir directive to `k:\math\include`. Then, the list file would only have to specify the archive name.

You may want to create a file in the main project directory that contains references to the individual list files. Such a file might look like this:

DOS
```
@math\localfil.lst
@dac\localfil.lst
@uio\localfil.lst
@rpt\localfil.lst
```

UNIX
```
@math/localfil.lst
@dac/localfil.lst
@uio/localfil.lst
@rpt/localfil.lst
```

If this file is called MASTFILE.LST, you can then use it in Version Manager commands as follows:

```
vcs -v"revised testing" @mastfile.lst
```

The command above assigns the version label `revised testing` to the tip revision of all of the archives contained in the included list files.

Using Group Files

Locate list files automatically

Group files are list files that Version Manager looks for using the search algorithm described in "How Version Manager Searches for Archives" on page 15. Group files could be placed in archive directories so that users would not have to explicitly state the path to the file.

Version Manager looks for all list files with the extension .GRP in the current directory and in directories specified by the "VCSDir directive" on page 175. If you use a group file and it is located in the current directory or in a VCSDir directory, you don't have to include its path or extension.

Special Considerations

- If you use the `@` command-line option without the *`list_file`* parameter, Version Manager expects input from the standard input device. You can redirect the output of another command to standard input so that Version Manager can accept additional command-line parameters from it.
- **For UNIX users**: UNIX shell escaping should not be used in list files.

Related Topics

Commands that can use list files...	
"GET" on page 59	"VCS" on page 100
"IDENT" on page 67	"VDEL" on page 114
"MAKEDB" on page 68	"VDIFF" on page 119
"REGEN" on page 87	"VJOURNAL" on page 129
"PUT" on page 73	"VLOG" on page 134
"READDB" on page 83	"VMRG" on page 141
"VCOMPRES" on page 91	"VSQL" on page 148
"VCONFIG" on page 94	

Specifying Dates and Times

Identify revisions by date and time

Several Version Manager command-line options require you to specify a date and time, or a range of dates and times.

Date/time format

Use the following rules when specifying dates and times:

- Specify the day of the month as a number between 1 and 31. If you also specify the month numerically, specify the day after the month.
- You can specify a month by its full name, any unique portion of its name, or a number between 1 and 12.
- Version Manager assumes that any number in the ranges of 80–99 and 1980–2035 is a year.
- Specify the time in the following form:

  hh[:mm[:ss]]

  - `hh` is the hour, `mm` is the minute, and `ss` is the second.
  - The form `hh` can be followed by a time indicator: `A`, `AM`, `P`, or `PM`. If you don't use a time indicator, the program assumes you are using 24-hour (military) time.
  - In the `hh:mm` and `hh` forms, the missing components default to zero.
- Time indicators and month names are not case-sensitive.
- Instead of colons, you can use tabs, commas (,), periods (.), slashes (/), or hyphens (-) to separate adjacent components. If you use tabs, enclose the date in quotation marks. Any other character indicates the end of the date/time specification.

Examples

The following are examples of date/time formats:

98-9-20-15:52	"9/20/98/4PM"
98/9/20/15:52	2035/12/31/15:52

NOTE All of the above date/time specifications may be equivalent, depending on the current date and time when the command is executed.

Using Date and Time Ranges

Establish ranges

Some Version Manager command options let you specify a range of dates and times. The earliest date you can specify is January 1, 1980; the latest is December 31, 2035.

Use the following formats to specify ranges:

- Implied range: Use a single date/time to imply a range based on the missing components.

Syntax `date`

- Dates before: Insert an asterisk before the specification to indicate dates/times prior to and including the specified date/time.

Syntax `*date`

- Dates after: Insert an asterisk after the specification to indicate dates/times beginning with and later than the specified date/time.

Syntax `date*`

- Explicit range: Specify both dates/times separated by an asterisk to define an inclusive range of dates.

Syntax

*first_date*second_date*

Defaults

The following table lists the values used if you omit a portion of a date/time range:

If you specify...	This value is used...
Nothing	All day today
*	From 00:00:00 1/1/1980 to 23:59:59 12/31/2035
The time only	That time today
The date only	00:00:00 to 23:59:59 on that day
The day only	That day in the current month and year
The month only	That month in the current year
The year only	That year

NOTE The date format determines whether ambiguous numbers are months and days, as what is defined first is used. If the date format is set to mm/dd/yy, Version Manager first assumes that the undefined number is a month. If the number is greater than 12 and less than the last day of the current month, it cannot be a month and is assumed to be a day of the current month. If the number is greater than 79, it is assumed to be a year.

Examples

The following examples below show the value entered, followed by the range it specifies. In the examples, the current date/time is April 15, 1998 14:35:22. Dates are represented as dd/mm/yy.

This entry...	Is equivalent to this range...
98	Jan 01 1998 00:00:00 * Dec 31 1998 23:59:59
Mar	Mar 01 1998 00:00:00 * Mar 31 1998 23:59:59
Apr/98	Apr 01 1998 00:00:00 * Apr 30 1998 23:59:59
Mar/11/98	Mar 11 1998 00:00:00 * Mar 11 1998 23:59:59
Apr/15/98	Apr 15 1998 00:00:00 * Apr 15 1998 23:59:59
Mar/11/98 4:23p	Mar 11 1998 16:23:00 * Mar 11 1998 16:23:59
96*98	Jan 01 1996 00:00:00 * Dec 31 1998 23:59:59
Mar/96*98	Mar 01 1996 00:00:00 * Dec 31 1998 23:59:59
*98	Jan 01 1980 00:00:00 * Dec 31 1998 23:59:59
98*	Jan 01 1998 00:00:00 * Dec 31 2035 23:59:59
May/23/98 4pm *	May 23 1998 16:00:00 * Dec 31 2035 23:59:59
4pm	Apr 15 1998 16:00:00 * Apr 15 1998 16:59:59
01-30-1998*02-10-2000	Jan 1 1998 00:00:00 * Feb 10 2000 23:59:59
15/2000	Jan 15 2000 00:00:00 * Dec 15 2000 23:59:59

Special Considerations

- You can use the DateFormat directive to change the order in which you enter the month, date, and year, and to specify the characters used to separate dates and times.
- By default, Version Manager recognizes the English names of the months. You can use the MonthNames directive to specify other names.

Related Topics

For information about...	See...
Commands to specify dates	"GET command" on page 29
Commands to specify date ranges	"VCS command" on page 56 "VLOG command" on page 76
Using special characters in syntax	"Escape Characters in Specifications" on page 16
Modifying the default date format	"DateFormat directive" on page 134
Using different names for the months	"MonthNames directive" on page 160

The remainder of this chapter contains a detailed description of each command in alphabetical order.

Accessing a Version Manager File Server

If an access control database or LDAP authentication is associated with the Client Name, you must use the `-id` option to present a user ID and password for validation.

-id **`-id`***`user_id:user_password`*

Specifies a user ID and password to authenticate against the access control database and/or LDAP server, if any, that is associated with the relevant file server path map in the Version Manager File Server Administration utility.

NOTE The `-id` option does not affect the user ID used to perform the CLI operation itself. The user ID for the CLI operation is determined in the normal manner.

By default, Version Manager checks for a user ID and password authentication source in the following order:

1. **MERANT_LOGIN_PRIORITY:** An environment variable. Valid values are: "`PCLI_ID`", "`EVENTUSERID`", and "*`user_id:user_password`*".
2. **-id:** The value passed through the `-id` command-line option. Its value is in the form of "*`user_id:user_password`*".
3. **PCLI_ID:** An environment variable. Its value is in the form of "*`user_id:user_password`*".
4. **EVENTUSERID:** An environment variable in the form of "*`user_id`*" used in combination with the EVENTPASSWORD variable (which contains a limited-lifetime encrypted password). Values for these variables are generated when an external command is executed from an Event Trigger, a Toolbar Command, or when using the PCLI `run -e` command.

Special Considerations

- Version Manager validates against the first authentication source that contains a valid value ("`PCLI_ID`", "`EVENTUSERID`", or "*`user_id:user_password`*"). It does NOT check subsequent authentication sources even if the initial one fails.

- You can cause PCLI_ID or EVENTUSERID to be the first authentication source that is checked by assigning the value "PCLI_ID" or "EVENTUSERID" to the environment variable MERANT_LOGIN_PRIORITY.
- You can pass a "hardcoded"user ID and password to any CLI command by assigning a "*user_id*:*user_password*" value to the environment variable MERANT_LOGIN_PRIORITY. No other authentication sources will be checked. NOTE You must include the colon (:) even if no password is required.

For information on setting up client access to a Version Manager File Server, see the *Administrator's Guide*.

FILT command

Provide a language-independent text filter system

Use the FILT command to copy certain parts of an input text file to an output file. To determine which text to include, you use conditional commands called *filter controls* to evaluate variables called *filter keys,* which you define in the input file or on the command line.

The input file can contain any type of text, so that you can use the same filter controls on different types of files, such as C code, assembler code, or help text.

Here are some circumstances in which the FILT command is useful:

- The FILT command provides a primitive preprocessor for source code written in a language that does not offer a preprocessor, such as COBOL.
- You can use the FILT command as a tool for managing ordinary text files, such as documentation or README files.
- The FILT command assists you in supporting separate lines of development in a single file. Use it to avoid certain problems that may be inherent in the maintenance of branches.

Exit code

FILT returns an exit code of 0 if successful, 1 otherwise.

Syntax

```
filt [option...] [input_file | -] [output_file]
```

Use the *input_file* parameter to specify the file to be filtered. Omit this parameter to redirect input from another command.

Use the *output_file* parameter to specify the file name used for the filtered text. Omit this parameter to display output on the screen.

To redirect input from another command and save filtered text to a file, use a hyphen (-) for the *input_file* parameter, and use the *output_file* parameter to specify the output file.

Options

-c Makes filter keys case-sensitive. By default, they are not case-sensitive.

-e `-Ex`

Defines the filter control character, *x*. The default is the percent sign (%).

-id **-id***user_id*:*user_password*

(File server only.) Specifies a user ID and password to authenticate against the access control database and/or LDAP server, if any, that is associated with the relevant file server path map in the Version Manager File Server Administration utility.

NOTE The -id option does not affect the user ID used to perform the CLI operation itself. The user ID for the CLI operation is determined in the normal manner.

-k -k*key*[,*key*...]

Specifies the filter keys that are assigned a value of true. Any filter key not specified with the *key* parameter is assigned a value of false. Separate filter keys with commas.

-p -p*column*

Specifies the column where the escape character occurs. The default is column 1.

-v Designates all environment variables as filter keys.

Using Filter Controls

Include text under certain conditions

Insert filter controls in the input text file to determine which portions of the file are included in the filtered output.

Syntax

```
%IF expression
        text
        .
        .
        .
%ELSEIF expression...
        text
        .
        .
        .
%ELSE expression
        text
        .
        .
        .
%END | %ENDIF
```

The filter controls are listed below:

Use this filter control...	To do this...
%IF	Include text if a condition is true.
%ELSEIF	Include text if the condition tested by the preceding filter control is false.
%ELSE	Include text if all other conditions in the conditional statement are false.
%END %ENDIF	End a conditional statement.
%DEFINE	Define a filter key.

Use this filter control...	To do this...
%UNDEFINE	Undefine a filter key.
%INCLUDE *file_name*	Insert the contents of *file_name* at the point of the %INCLUDE control. You can nest included files up to 20 levels deep.

- By default, the filter control character, %, must be in column one. You can redefine the filter control character and its location using the -e and -p command-line options.
- To extend lines containing filter controls across multiple lines, end all but the last line with a backslash (\).
- You can nest filter controls to a depth limited by the stack space. By default, this is 2048 bytes, which allows about 100 levels of nesting.
- You can insert one or more spaces between the filter control character and the filter control word (IF, ELSE, etc.). This lets you indent nested filter controls to reflect the nesting level.

Using Filter Expressions

Variables used in expressions

The filter expressions consist of *filter keys* and *operators.* Filter keys are variables which you can define on the command line or in the input file. Filter keys must begin with a letter or underscore (_) and can be followed by additional letters, digits, or underscores (100 characters maximum per filter key).

Operators are listed below in order of decreasing precedence. Use parentheses in expressions to force associations.

!	Logical NOT
&	Logical AND
|	Logical OR

Example

The following example shows how to control the contents of an output file by defining filter key named XENIX on the command line.

```
%IF XENIX
     (Text of the XENIX READ.ME file.)
%ELSE
     (Text of the MS-DOS READ.ME file.)
%END
```

Special Considerations

- FILT distinguishes between spaces and tabs, so be consistent in their use.
- **For UNIX users**: The FILT command is not distributed with the UNIX version of Version Manager.

Related Topics

For information about...	See...
Redirecting error messages	"RSE command" on page 49
Other Version Manager commands	"Using Commands" on page 8

GET command

Retrieve a revision

Use the GET command to check out revisions from archives.

Privileges required. The GET command requires the Get privilege, GetTip privilege, or GetNonTip privilege.

Exit code

GET returns an exit code of 0 if successful, 1 otherwise.

Syntax

```
get [option...] file_name...
```

Frequently used commands

To do this...	Use this command...
Check out a read-only workfile	`get`
Check out a locked workfile	`get -l`
Check out a specific revision	`get -fr -r`
Check out a revision based on its version label	`get -v`
Check out a revision based on its time/date stamp	`get -d`
Display output instead of creating a workfile	`get -p`
Redirect output	`get -x`
Check out a writable workfile	`get -w`

Default

If not otherwise specified, the search path includes the current directory for archives and workfiles. Refer to the VCSDir directive for more details on changing the default behavior.

Overwrite writable workfiles

If a writable file exists with the same name as the specified workfile, the program prompts for permission to overwrite the file. Use the `-q` or `-y` options to give advance permission. Use the `-n` option to deny permission and override permission granted by other options. If you are using the `-q` option but want to be prompted for permission to overwrite, use the `-qo` option.

Version Manager timestamps workfiles with the time and dates they were last modified. When you check out a file, you can use the `-t` option reset the timestamp to reflect the current time and date.

Specifying Files

Specify archives and directories

In addition to the methods of specifying files outlined in "Specifying File Names" on page 10, you can specify an archive plus a directory for the retrieved workfile, as shown below:

DOS

```
get archive([drive:]\directory)
```

UNIX

```
get archive"(directory)"
```

You can use wildcard characters in this type of archive specification. See "Specifying Names Using Wildcards" on page 13 for more information.

You can use environmental variables to change the separator character the command line uses to separate archive and workfile names (or archive and directory names) in a command. See "Specifying Both Workfiles and Archives" on page 11 for more information.

Example The following example specifies the archive and directs GET to place the resulting workfile in F:\TESTDIR for Windows and */testdir* for UNIX:

DOS
```
get test.c_v(f:\testdir)
```

UNIX
```
get test.c_v"(/testdir)"
```

or
```
get test.c_v\(/testdir\)
```

Default versions The DefaultVersion directive specifies a version label to use if no revision is specified for a command. It affects the following commands:

get -l get -p get -w get -t get -v

You can override this directive by using a hyphen (-) after any of these command-line options. When you override the DefaultVersion directive, you check out the tip revision of the trunk.

Options

@ `@[list_file]`

Use the `@` option to read *`list_file`* for additional command-line options. Version Manager reads additional options from the file before reading the rest of the command line. Use the `@` option with no parameter if you're redirecting input from another command. See "List and Group Files" on page 21 for more information.

If the file has the extension .GRP, you can omit the extension and the path. Version Manager searches for it in the directories specified by the VCSDir directive.

-# `-#code`

Use the `-#` option to display diagnostic information for debugging purposes. The *`code`* parameter is a number representing diagnostic information according to the following list:

`1`	Display each line of the configuration file as it is processed.
`2`	Display internal errors.
`4`	Toggle the sign-on message.
`500`	Display the archive owner, access list, access control database name, current user ID, and their privileges. The currently active privileges are in uppercase letters.

To specify more than one diagnostic item, add the numbers together. For example, use `-#3` to display the diagnostic information represented by numbers `1` and `2`.

The -# option should be the first option after the command. Version Manager processes commands from left to right and does not display debugging information until it reaches the -# option.

-c `-c[- | configuration_file | directory_name]`

Instructs Version Manager to look for configuration information in a place other than in the master configuration file or the default configuration file.

See the "Configuration Files" on page 111 topic for details on using the -c option.

-d -d*date/time*

Checks out the last revision that was checked in before *date/time*. This option is useful only if you use neither the -r nor the -v option, or if you use the -r option with a branch number. If you use neither the -r nor the -v option, you retrieve the latest qualifying trunk revision. If the -r option specifies a branch number, you retrieve the latest qualifying revision on the branch. This option uses the stored "Check in" time and date and does *not* use the "Last modified" time and date for its comparison (refer to -u on 32)

-fg -fg

Looks up revisions associated with the promotion group defined by the -g parameter. The -g parameter is required; an error occurs if it is not.

-fr -fr

Looks up a specific revision based on the revision specified by the -r parameter. If -r is not specified, then the tip or default revision is retrieved.

-g -g*promotion_group*

Checks out a revision by promotion group. This is required if the -fg parameter is set. If Version Manager can't find a revision associated with the promotion group you specify, it will check out the revision associated with the next highest group in the promotion model.

For example, this option is useful during application builds in which you want to check out all files that have passed a certain benchmark, such as a quality assurance review, or higher in preparation for a compile.

-h Displays help for command-line options. The program terminates after it processes the -h option even if you specify other options.

-id **-id***user_id*:*user_password*

(File server only.) Specifies a user ID and password to authenticate against the access control database and/or LDAP server, if any, that is associated with the relevant file server path map in the Version Manager File Server Administration utility.

NOTE The -id option does not affect the user ID used to perform the CLI operation itself. The user ID for the CLI operation is determined in the normal manner.

-l -l[*revision_number* | *version_label* | *promotion_group*]

Locks the specified revision with the current user ID. If the specified revision is already locked, or if you already have another revision locked and MultiLock is not in effect, Version Manager terminates the GET and issues an error.

If you don't specify a revision, you retrieve the latest trunk revision, unless the DefaultVersion directive is in effect. If you specify a branch number, you retrieve the latest revision on the branch.

If you're using the MultiLock directive, Version Manager displays the check-in revision number that is reserved for this revision. No other user can check in a workfile to this revision number in this archive unless the number is released.

If you're using a promotion model and the revision is not currently assigned to a development group, the program automatically assigns the revision to the development group before checking it out. If more than one development group exists, the program prompts you to specify the development group.

Privileges required. This option requires the Lock privilege, LockTip privilege, or LockNonTip privilege.

-n Denies permission to overwrite existing writable workfiles. See also -qn.

-p Displays output instead of creating a workfile.

-q Selects the quiet mode of operation, in which Version Manager only reports error conditions. The -q option gives permission in advance to overwrite existing workfiles.

-qo Directs Version Manager to report only error codes, but asks permission before overwriting existing workfiles. When you use -qo with the -y option, it is equivalent to -q.

-qn Denies permission to overwrite existing workfiles. This option has the same effect as using the -n option with either the -q or the -qo option. See -n.

-r -r[*revision_number* | *version_label* | *promotion_group*]

Specifies the revision number, version label, or promotion group to be checked out. To check out the latest revision with that major number, specify the branch revision number. To check out the latest revision on a branch, specify a version label to check out the corresponding revision.

If you use -r without a revision number, a promotion group or a version label, the program uses the latest trunk revision, unless the DefaultVersion directive is defined. If the version label begins with a number, precede it with a backslash (\).

-s -s*suffix_template*

Specifies the archive suffix template to use instead of the current value of ArchiveSuffix.

-t -t[*revision_number* | *version_label* | *promotion_group*]

Sets the time and date stamp of the checked-out file to the current time. Otherwise, the file carries the date and time it was last modified. Using this option is equivalent to using the Configuration Builder TOUCH command on the checked-out file.

-u -u[*date/time*]

Checks out a revision on the trunk only if it has a later date and time than the existing workfile, if any. This is useful for updating a set of workfiles with the latest trunk revisions. If you supply the optional *date/time* parameter, Version Manager only checks out a revision if it is newer than the *date/time* specified. This option uses the "Last Modified" date and time. See the -d option.

To specify a revision on a branch, you must use the -r option.

-v -v*version_label*

Specifies a version to be checked out.

-w -w[*revision_number* | *version_label* | *promotion_group*]

Checks out a writable workfile. This option does not affect the lock status.

-xe -xe*file_name*

Redirects all Version Manager status, program, and error messages to *file_name*.

-xo -xo*file_name*

Redirects standard output to *file_name*.

-xo+e `-xo+e`*`file_name`*

Redirects standard output and error messages to *`file_name`*.

-xrenumber `xrenumber = column_start-column_end [from start by number]`

Specifies the column range that the program renumbers when retrieving the workfile. The *`column_start`* and *`column_end`* parameters specify column numbers. Column numbering begins with column number 1. *`start`* is the number the beginning number, and *`number`* is the amount by which to increase this number for each line. Version Manager fills line numbers with zeros beginning on the left (000010, 000020, etc.)

This option overrides the Renumber directive.

-y Answers yes in advance to any question the program may ask you. The effect is similar to that of the `-q` option, except that `-q` also suppresses some informative output.

Examples

- The following example checks out the latest revision of TEST.C from the trunk and locks it.
  ```
  get -l test.c
  ```
- The following example checks out the latest trunk revision from each of the archives that contains C source code and locks them.
  ```
  get -l *.c_v
  ```
- The following example checks out the revisions associated with the version label *Release 1* and locks them.
  ```
  get -l"Release 1" *.??v
  ```
- The following example checks out the latest revision of PROG.PAS from branch 1.2.1 and locks it.
  ```
  get -l1.2.1 prog.pas
  ```

Related Topics

For information about...	See...
Specifying dates and times	"DateFormat directive" on page 134
How Version Manager translates suffixes	"ArchiveSuffix directive" on page 122
Defining a default revision or version	"DefaultVersion directive" on page 135
Renumbering lines	"Renumber directive" on page 167
Using wildcard characters	"FirstMatch directive" on page 151

IDENT command

Display all Version Manager keywords

Use the IDENT command to display Version Manager keywords in files. IDENT can display keywords in any type of file (this command works with a workfile, Archive revision, flat file, object file, or executable file).

Exit code

IDENT returns an exit code of 0 if successful, 1 otherwise.

Syntax `ident [option...] [file_name...]`

If you omit the *`file_name`* parameter, IDENT obtains the file to scan from STDIN, if STDIN has been redirected.

Options

@ `@[list_file]`

Use the `@` option to read *`list_file`* for additional command-line options. Version Manager reads additional options from the file before reading the rest of the command line. Use the `@` option with no parameter if you're redirecting input from another command.

If the file has the extension .GRP, you can omit the extension and the path. Version Manager searches for it in the directories specified by the VCSDir directive.

-h Displays help for command-line options. The program terminates after it processes this option even if you specify other options.

-id **`-id`***`user_id`*`:`*`user_password`*

(File server only.) Specifies a user ID and password to authenticate against the access control database and/or LDAP server, if any, that is associated with the relevant file server path map in the Version Manager File Server Administration utility.

NOTE The `-id` option does not affect the user ID used to perform the CLI operation itself. The user ID for the CLI operation is determined in the normal manner.

-xo `xo`*`file_name`*

Redirects standard output to *`file_name`*.

Examples

- The following command stores a record of all keywords in files with the extension .H_V in the file KEYWORDS.OUT:
  ```
  ident -xokeywords.out *.h_v
  ```
- The following command checks out a read-only copy of all revisions associated with the version label `Alpha`, then expands the keywords in the revisions that were retrieved.
  ```
  get -vAlpha -p *.??v | IDENT
  ```

Special Consideration

The IDENT command only displays the values of keywords if the ExpandKeywords directive was in effect when the archive was created.

Related Topics

For information about...	See...
Enabling keyword expansion	"ExpandKeywords Touch directive" on page 150
Other Version Manager commands	"Using Commands" on page 8
Keywords	*Administrator's Guide*

MAKEDB command

Create the access control database

Use the MAKEDB command to create an access control database file from a text file. If you are using the MAKEDB on the output of the READDB command, you must edit the text file to move the users to the bottom of the file.

NOTE To use this command with a Version Manager File Server, the Enable Utilities option must be correctly set in the Version Manager File Server Administration utility.

Specify the database file name

Before you create or change the access control database, you must do one of the following to specify its name:

- Embed the name of the database file in the MAKEDB.EXE executable using the VCONFIG -A command.
- Specify the file name with the AccessDB directive.
- Specify the file name as a parameter to the MAKEDB -A command.

If you don't identify the name of the access control database using one of these three methods, MAKEDB issues an error message.

NOTE If readdb is used on the access control database that has an expiration date, makedb cannot read the output of the readdb command. The expiration date in the access.txt file cannot be read by makedb.

Exit code

MAKEDB returns an exit code of 0 if successful, 1 otherwise.

Syntax

```
makedb [options...] [file_name]
```

Use the *file_name* parameter to specify the name and location of your access control text file. Omit this parameter to redirect input from another command.

Frequently Used Commands

To do this...	Use this command...
Specify the name of the access control database	MAKEDB -A
View each record as it is built	MAKEDB -V

Options

The following lists options for the MAKEDB command and how they are used:

@ `@[list_file]`

Use the `@` option to read *list_file* for additional command-line options. Version Manager reads additional options from the file before reading the rest of the command line. Use the `@` option with no parameter if you're redirecting input from another command.

If the file has the extension .GRP, you can omit the extension and the path. Version Manager searches for it in the directories specified by the VCSDir directive.

-# `-#code`

Use the -# option to display diagnostic information for debugging purposes. Use the *code* parameter to select one of the following diagnostic functions:

1	Display each line of the configuration file as it is processed.
2	Display internal errors.
4	Toggle sign-on message.
500	Display the archive owner, access list, access control database name, current user ID, and assigned privileges. The currently active privileges display in uppercase letters.

To specify more than one diagnostic item, add the numbers together. For example, use –#3 to display the diagnostic information represented by numbers 1 and 2.

The –# option should be the first option after the command. Version Manager processes commands in left-to-right order and does not display debugging information until it reaches the –# option.

-a –a*database_name*

Specifies the name of the resulting access control database. If you don't use this option, MAKEDB uses the file name that was embedded in it using the VCONFIG -A command or the one provided by the AccessDB directive. A value assigned to the AccessDB directive overrides a value embedded in MAKEDB.EXE by VCONFIG -A.

-c –c[- | *configuration_file* | *directory_name*]

Instructs Version Manager to look for configuration information in a place other than in the master configuration file or the default configuration file.

See "Configuration Files" on page 111 for details on using the –c option.

-h Displays help for command-line options. The program terminates after it processes this option even if you specify other options.

-i Creates the access control database and forces the addition of an Admin user, who is assigned the SuperUser privilege by default.

-id **–id***user_id*:*user_password*

(File server only.) Specifies a user ID and password to authenticate against the access control database and/or LDAP server, if any, that is associated with the relevant file server path map in the Version Manager File Server Administration utility.

NOTE The –id option does not affect the user ID used to perform the CLI operation itself. The user ID for the CLI operation is determined in the normal manner.

-q Suppresses warning and sign-on messages.

-v Displays each file as it is built.

-xe -xefile_name

Redirects status, program, and error messages to *file_name*.

-xo –xo*file_name*

Redirects standard output to *file_name*.

-xo+e –xo+e*file_name*

Redirects standard output and error messages to *file_name*.

Examples

- The following example creates the database whose name is embedded in the MAKEDB executable from the text file C:\SERENA\ACCESS.TXT for Windows and */usr/serena/access.txt* for UNIX, and displays each record as it is built:

DOS

```
makedb -v c:\serena\access.txt
```

UNIX

```
makedb -v /usr/serena/access.txt
```

- The following example creates the database C:\SERENA\ACCESS.DB from the text file C:\SERENA\ACCESS.TXT for Windows, and creates */usr/serena/access.db* from the text file */usr/serena/access.txt* for UNIX.

DOS

```
makedb -ac:\serena\access.db c:\serena\access.txt
```

UNIX

```
makedb -a/usr/serena/access.db /usr/serena/access.txt
```

Special Considerations

- For security reasons, this command is not placed in the same directory as other commands when Version Manager is installed. By default, it is installed in the `\Bin\Admin` subdirectory where you installed Version Manager.
- After using MAKEDB to create the access control database, remove the original text file. If you redirected MAKEDB output to a file, remove that file as well, because it contains user passwords.

  Store these files in a network directory that can only be accessed by the Version Manager administrator, or on a separate floppy disk in a secure location.
- Control access to the READDB and MAKEDB commands. Anyone who can use these commands can read or modify all access control database information, including passwords.
- If a privilege set name contains a dash character (-) , you must quote the privilege set name when assigning it. Do this by manually editing the output of the READDB command before running the MAKEDB command against the output. For example, change: `USER bsmith/ (Read-Only)` to: `USER bsmith/ ("Read-Only")`.

Related Topics

For information about...	See...
Viewing the contents of the access control database	"READDB command" on page 44
Specifying the name of the access control database	"VCONFIG command" on page 52
Other Version Manager commands	"Using Commands" on page 8

PRINTENV command

Display environment variables

Use the PRINTENV command to display the current values of environment variables.

Exit code

PRINTENV returns an exit code of 0 if it finds environment variables defined. It returns an exit code of 1 if there are no variables defined. It returns an exit code of 2 if there is insufficient memory for it to operate.

Syntax

`printenv [variable...]`

To display only certain variables, use the *variable* parameter. You can use wildcard characters to display variables that match a wildcard pattern.

Special Consideration

For UNIX users: The PRINTENV command is not available for the UNIX version of Version Manager.

Related Topics

For information about...	See...
Other Version Manager commands	"Using Commands" on page 8

PUT command

Add a revision to an archive

Use the PUT command to store a new revision in an archive. If the archive does not exist, PUT creates it, unless the NoAutoCreate directive is in effect. If lock checking is enabled, you must be the user who locked the preceding revision in order to check in a new revision, unless the MultiLock directive is in effect.

When the PUT command is used to check in subsequent revisions, Version Manager checks whether the new revision is different from the previous revision. If the two are identical, the program asks whether you want to store the revision anyway. A different timestamp is not a basis for Version Manager deciding that differences exist between the new revision and the previous revision.

Unless you specify otherwise, PUT creates a new revision number when you check in a file by incrementing the minor number of the previous revision. When you want to use a new major number, specify the new revision number using the `PUT -R` command. The revision number can be any number greater than the last revision.

The PUT command also checks the timestamp on the file you are checking in. If it is older than its predecessor, the program prompts for permission to continue.

Privileges required. Using the PUT command requires allowing the Put privilege, PutBranch privilege, or PutTrunk privileges; creating new archives requires the InitArchive privilege.

Exit code

PUT returns an exit code of 0 if successful, 1 otherwise.

Syntax

`put [option...] file_name...`

Frequently Used PUT commands.

To do this...	Use this command...
Check in a workfile	put
Check in a workfile even if it is unchanged	put -f
Check the workfile in and immediately out with a lock	put -l

To do this...	Use this command...
Assign a revision number	put -r
Assign a version label	put -v
Specify a change description	put -m
Specify a workfile description	put -t
Check the workfile in and immediately out without a lock	put -u

Default versions

The DefaultVersion directive affects the following commands:

put -l
put -v

This directive can be overridden by using a hyphen (-) after the option.

Applying Change Descriptions

Use the same description for multiple files

When a new revision is checked in, the PUT command prompts for a description of the changes. You can type a description or call the description from a file.

If a change description is typed for the first file, Version Manager displays the following message for each subsequent file:

```
Use previous comment for file "file_name" ? (y/n)
```

As with other Version Manager prompts, this prompt can be answered in advance with the `-y` or `-n` option.

The same change description can be applied to more than one file by placing the description of the changes in a file and specifying that file using the `-m` option.

If an unchanged revision is stored, the program does not prompt for a change description. Instead, it uses the description *No change*, unless other text is supplied using the `-m` option.

Specifying a Revision Number

Store a specific revision number

If you specify a revision number when storing a new revision, the number must be consistent with the revision that was locked (branch vs. trunk). For example, if revision 1.2.3.1 is locked, Revision 1.2.4.1 can't be specified. The new revision must be on the same branch (1.2.3) as the locked revision.

However, you can specify a higher revision number for a trunk revision. For example, if the latest trunk revision (such as 6.10) is locked, a revision with a higher major number (such as 7.1) can be specified.

If a revision number is not specified, PUT assigns the next revision number after the revision that is currently locked, if possible. If the next revision number is already used and lock checking is enabled, the program creates a branch. If lock checking is not enabled, the revision becomes the trunk's tip revision.

Options

@ `@[list_file]`

Use the `@` option to read *list_file* for additional command-line options. Version Manager reads additional options from the file before reading the rest of the command line. Use the `@` option with no parameter if you're redirecting input from another command.

If the file has the extension .GRP, the extension and the path can be omitted. Version Manager searches for it in the directories specified by the VCSDir directive.

-# `-#code`

Use the -# option to display diagnostic information for debugging purposes. Use the *code* parameter to select one of the following diagnostic functions:

`1`	Display each line of the configuration file as it is processed.
`2`	Display internal errors.
`4`	Toggle the sign-on message.
`500`	Display the archive owner, access list, access control database name, current user ID and their privileges. The currently active privileges are in uppercase letters.

To specify more than one diagnostic item, add the numbers together. For example, use `-#3` to display the diagnostic information represented by numbers `1` and `2`.

The -# option should be the first option after the command. Version Manager processes commands in left-to-right order and will not display debugging information until it reaches the -# option.

-a `-a[user_id[,user_id...]]`

Specifies an access list when using PUT to create an archive. Separate multiple user IDs with commas. Omit the *user_id* parameter to create archives with an empty access list.

The access list you specify supersedes any access list specified with the AccessList directive. This option has no effect on existing archives.

-b Instructs Version Manager to ignore leading and trailing white space when determining whether the file has changed. White space characters include spaces, tabs, carriage returns, line feeds, and form feeds. This option does not affect the change information stored in the archive.

-c `-c[- | configuration_file | directory_name]`

Instructs Version Manager to look for configuration information in a place other than in the local configuration file or the default configuration file.

See "Configuration Files" on page 111 for details on using the `-c` option.

-f Stores the revision even if it is identical to its predecessor. Otherwise, PUT indicates that the checked-in revision is the same and asks for confirmation before continuing.

-fb Forces a branch even if the revision being modified is a tip revision. This is useful when checking in multiple workfiles that constitute a new line of development.

Privileges required. The `-fb` option requires the StartBranch privilege.

-g `-gpromotion_group`

Identifies a revision by its promotion group. Used primarily to create a new archive and assign it to a promotion group.

-h Displays help for command-line options. The program terminates after it processes this option even if other options are specified.

-id **-id***user_id*:*user_password*

(File server only.) Specifies a user ID and password to authenticate against the access control database and/or LDAP server, if any, that is associated with the relevant file server path map in the Version Manager File Server Administration utility.

NOTE The `-id` option does not affect the user ID used to perform the CLI operation itself. The user ID for the CLI operation is determined in the normal manner.

-l Checks in the revision and immediately checks it out with a lock. This option overrides the DeleteWork directive. See also `-u`.

Privileges required. The `-l` option requires the Lock privilege, LockTip privilege, or LockNonTip privileges.

-m Supplies a change description. If a parameter to this option is not specified, the program prompts for the description.

When the PUT command is used to create an archive, the change description `Initial revision` is used by default. The `-m` option can be used to override this. See also `-t`.
`-m`*text*

Use this option with a *text* parameter to specify the change description on the command line. If *text* contains spaces, enclose it in double quotation marks ("). If *text* begins with an at sign (`@`), precede it with a backslash (\).
`-m@`*file_name*

Use the `-m` option with the *@file_name* parameter to read the change description from *file_name*.
`-m@`

Use this option with the `@` parameter to read the change description from the file whose name is computed from the workfile name using the value specified by the MessageSuffix directive.
`-m@`*directory*

Use this option with the *@directory* parameter to read the change description from the file located in the specified directory whose name is derived from the workfile.

-n Answers no in advance to queries issued by PUT. The possible queries are:

- Should the program check in unchanged workfiles?
- Should the program check in a file with a modification time older than its predecessor?
- Should the program delete a previous version label that is identical to the new version label?
- Should the program use the previous comment for a multiple file check in?

-q Selects the quiet mode of operation, in which Version Manager only reports error conditions. The `-q` option answers yes in advance to any questions PUT may ask.

-qo Selects the quiet mode of operation without suppressing any questions the program may ask. When you use `-qo` is with the `-y` option, it is equivalent to `-q`.

-r `-r revision_number | version_label | promotion_group`

Assigns a revision number to a revision you are checking in. The revision number you specify must be numerically higher than the highest existing revision number.

If you specify only the major part of a revision number, Version Manager uses the next available minor number.

Privileges required. If you are starting a branch, `put -r` requires the StartBranch privilege in addition to the PutBranch privilege.

-s `-ssuffix_template`

Specifies the archive suffix template to use instead of the current value of ArchiveSuffix.

-t Specifies the workfile description when creating an archive. PUT ignores the `-t` options if they are used for existing archives, unless there is no workfile description in the archive. See also -m.

If a parameter to the `-t` option is not specified, the program prompts for the description.
`-ttext`

Use the `-t` option with a *text* parameter to specify the workfile description on the command line. If *text* contains spaces, enclose it in double quotation marks ("). If *text* begins with an at sign (`@`), precede it with a backslash (\).
`-t@file_name`

Use the `-t` option with the *@file_name* parameter to read the workfile description from *file_name*.
`-t@`

Use the `-t` option with the `@` parameter to read the workfile description from the file whose name is computed from the workfile name using the value specified by the MessageSuffix directive.
`-t@directory`

Use the `-t` option with the *@directory* parameter to read the workfile description from the file located in the specified directory whose name is derived from the workfile.

-u Checks in the revision and immediately checks it out without a lock (the file is checked out as a read only file). This option overrides the DeleteWork directive. See also `-l`.

-v `-vversion_label`

Associates *version_label* with a revision you are checking in. If the *version_label* is already assigned to another revision, Version Manager prompts for permission to remove it. Use either the `-q` or `-y` options to provide permission, or the `-n` option to deny permission. The `-n` option has precedence.

Privileges required. The `-v` option requires the ModifyVersion privilege if the version label already exists; otherwise, it requires the AddVersion privilege.
`-vfloating_version:*`

Use this option with the *floating_version* parameter to assign a floating version label to the revision being stored. Floating version labels "float" with the tip of the branch or trunk to which they are assigned.

-xcolumnmask `-xcolumnmask = "start-end [(numeric)]..."`

Changes the columns that Version Manager converts to spaces when generating change information. This option overrides the ColumnMask directive.

The *start* and *end* parameters specify column numbers. Column numbering begins with column number 1. More than one column range can be defined on the command line, and they can be specified in any order. Ranges can overlap. To turn off column masking for the archive, use zeroes for the column range.

To restrict masking to numeric fields only, use the keyword (`Numeric`) following a column range. In this case, the program only masks a field if the first character of the field is numeric.

```
-xcolumnmask = cobol
```

This form of the `-xcolumnmask` option is equivalent to the following:

```
-xcolumnmask = "1-6 (numeric)" "73-80"
```

-xe `-xefile_name`

Redirects status, program, and error messages to *file_name*.

-xo `-xofile_name`

Redirects standard output to *file_name*.

-xo+e `-xo+efile_name`

Redirects standard output and error messages to *file_name*.

-xrecordlength `-xrecordlength = record_length`

Changes the record length of the workfile. The *record_length* parameter specifies the number of characters. The maximum record length is 64K. To turn off the record length, use zero. Use this option only with non-CRLF records with fixed length.

-y Answers yes in advance to any question the program may ask. The effect is similar to that of the `-q` option, except that `-q` also suppresses some informative output. This option gives implicit permission to overwrite any existing workfiles unless the `-n` option is also used. The `-n` option overrides the `-y` and `-q` options.

Examples

- The following example stores a new revision of TEST.C in its archive.
  ```
  put test.c
  ```
- The following example stores a new revision of the workfile associated with the archive TEST.C_V in that archive, leaving a copy of the new revision on disk for further updates. The revision is locked for exclusive update.
  ```
  put -l test.c_v
  ```
- The following example stores files with extension .C in their archives, and supplies a negative response to any queries.
  ```
  put -n *.c
  ```
- The following example stores C workfiles in their archives and assigns the version label `Alpha Test` to each new revision. The command supplies an affirmative response to any queries.
  ```
  put -v"Alpha Test" -y *.c
  ```

Related Topics

For information about...	See...
Using list files	"Using List Files" on page 21
Floating version labels	"Specifying Version Labels in Commands" on page 17
Determining whether the PUT command can create archives	"AutoCreate directive" on page 126
Defining a default revision or version	"DefaultVersion directive" on page 135
Using column masking	"ColumnMask directive" on page 129
How the -c option affects configuration file processing	"Configuration Files" on page 111
Specifying record length	"RecordLength directive" on page 165
Using wildcard characters	"Specifying Names Using Wildcards" on page 13 "FirstMatch directive" on page 151
Keywords	*Administrator's Guide*

READDB command

Display access control database records

Use the READDB command to display records in the access control database.

Privileges required. The READDB command requires the ViewAccessDB privilege. If no access control database is enabled, the READDB command is unrestricted. For information on deactivating the access control database, see "AccessControl directive" on page 117.

NOTE To use this command with a Version Manager File Server, the Enable Utilities option must be correctly set in the Version Manager File Server Administration utility.

The following is an example of a partial security report produced using `readdb -p`:

```
USER bruceb (UNLIMITED)
USER Admin (SUPERUSER)
    USER krisb (UNLIMITED)
USER kasiaj (SUPERUSER)
USER beckyp/rhino (UNLIMITED, Documentation)
USER ken (UNLIMITED)
USER betsyf (UNLIMITED)
USER georges (UNLIMITED)
GROUP TeamA (Developer): georges beckyp krisb bruceb
PRIVILEGE custom: \
        LOCKTIP \
        LOCKNONTIP \
        UNLOCK \
        BREAKLOCK \
        GETTIP \
        GETNONTIP \
        PUTTRUNK \
        PUTBRANCH \
        STARTBRANCH \
```

```
        CHANGEACCESSLIST \
        CHANGEOWNER \
        CHANGEPROTECTION \
        CHANGECOMMENTDELIMITER \
        CHANGEWORKFILENAME \
        MODIFYWORKFILEDESCRIPTION \
        MODIFYCHANGEDESCRIPTION \
.
.
.
```

Readdb lists:

- Each user name. Optionally, you can display the user names and passwords using the `-p` option.
- Each access list group.
- All default and custom privilege sets.

Each user definition line contains the user ID/password and the given privileges or privilege set in parentheses. See the `-p` option for further information on displaying passwords.

Each access list group definition line contains the name of the group, the given privileges or privilege set in parentheses, and the users assigned to the group.

Each privilege set definition line contains the name of the privilege set followed by the base privileges that define the privilege set.

Comment lines, if any, are preceded by either a pound sign or an exclamation point.

Exit codes

READDB returns an exit code of 0 if successful, 1 otherwise.

Syntax

`readdb [option...]`

Frequently used READDB commands

To do this...	Use this command...
Name the access control database	READDB -A

Options

@

`@[list_file]`

Use the `@` option to read *`list_file`* for additional command-line options. Version Manager reads additional options from the file before reading the rest of the command line. Use the `@` option with no parameter if you're redirecting input from another command.

If the file has the extension .GRP, you can omit the extension and the path. Version Manager searches for it in the directories specified by the VCSDir directive.

-#

`-#code`

Use the -# option to display diagnostic information for debugging purposes. Use the *code* parameter to select one of the following diagnostic functions:

1	Display each line of the configuration file as it is processed.
2	Display internal errors.
4	Toggle the sign-on message.
500	Display the archive owner, access list, access control database name, current user ID, and their privileges. The currently active privileges are in uppercase letters.

To specify more than one diagnostic item, add the numbers together. For example, use –#3 to display the diagnostic information represented by numbers 1 and 2.

The -# option should be the first option after the command. Version Manager processes commands in left-to-right order and will not display debugging information until it reaches the -# option.

-a –a*database_name*

Specifies the name of the access control database to be read. If you don't use this option, READDB uses the value embedded in it by the VCONFIG -A command or the name specified by the AccessDB directive. The value assigned to the AccessDB directive overrides the value embedded in READDB by VCONFIG -A.

-c –c[- | *configuration_file* | *directory_name*]

Instructs Version Manager to look for configuration information in a place other than in the master configuration file or the default configuration file.

See "Configuration Files" on page 111 for details on using the –c option.

-h Displays help for command-line options. The program terminates after it processes this option even if you specify other options.

-id **–id***user_id*:*user_password*

(File server only.) Specifies a user ID and password to authenticate against the access control database and/or LDAP server, if any, that is associated with the relevant file server path map in the Version Manager File Server Administration utility.

NOTE The –id option does not affect the user ID used to perform the CLI operation itself. The user ID for the CLI operation is determined in the normal manner.

-p Displays the current user's password. This is useful if you forget your password while you are still logged in. If you have the SuperUser privilege, this option displays all passwords in the database.

-xe –x*file_name*

Redirects status, program, and error messages to *file_name*.

-xo –xo*file_name*

Redirects standard output to *file_name*.

-xo+e –xo+e*file_name*

Redirects standard output and error messages to *file_name*.

Examples

- The following example reads the database file C:\SERENA\ACCESS.DB for Windows and */usr/serena/access.db* for UNIX:

DOS
```
readdb -ac:\serena\access.db
```

UNIX
```
readdb -a/usr/serena/access.db
```

- The following example reads the database file and displays the passwords:

```
readdb -p
```

- The following example shows how to use READDB output as input to MAKEDB. First, redirect the output of MAKEDB by typing:

DOS
```
readdb -ac:\serena\access.db > data.txt
```

UNIX
```
readdb -a/usr/serena/access.db > data.txt
```

Edit the file DATA.TXT with your text editor. Then use DATA.TXT as input to the MAKEDB command by typing:

DOS
```
makedb -ac:\serena\access.db data.txt
```

UNIX
```
makedb -a/usr/serena/access.db data.txt
```

Special Considerations

- For security reasons, this command is not placed in the same directory as other commands when Version Manager is installed. By default, it is installed in the `\Bin\Admin` subdirectory where you installed Version Manager.
- Control access to the READDB and MAKEDB commands. Anyone who can use these commands can read or modify all access control database information, including passwords.
- There is no default access control database name. To identify the access control database, use the VCONFIG -A command or the AccessDB directive.
- If a privilege set name contains a dash character (-) , you must quote the privilege set name when assigning it. Do this by manually editing the output of the READDB command before running the MAKEDB command against the output. For example, change: `USER bsmith/ (Read-Only)` to: `USER bsmith/ ("Read-Only").`

Related Topics

For information about...	See...
Creating an access control database	"MAKEDB command" on page 35
Configuring Version Manager to use the access control database	"AccessDB directive" on page 117 "VCONFIG command" on page 52
Other Version Manager commands	"Using Commands" on page 8

REGEN command

Regenerate files

Use the REGEN command to regenerate a file by applying editing instructions contained in a delta file to a reference file. Before using the REGEN command, create the delta file using the VDIFF -D command.

Using VDIFF and REGEN to encode and decode the changes to text files is useful when you want to minimize file size for storage or telecommunication.

We provide the C language source code for REGEN in a file named REGEN.C. You can use this source code as the basis for implementing REGEN under other operating systems.

Exit code REGEN returns an exit code of 0 if successful, 1 otherwise.

Syntax `regen [option...] reference_file delta_file [> target_file]`

Use the *reference_file* parameter to specify the file to which to apply the changes in the *delta_file*. If you omit the redirection symbol (>) and *target_file* parameter, REGEN displays its output.

Options

@ `@[list_file]`

Use the `@` option to read *list_file* for additional command-line options. Version Manager reads additional options from the file before reading the rest of the command line. Use the `@` option with no parameter if you're redirecting input from another command.

If the file has the extension .GRP, you can omit the extension and the path. Version Manager searches for it in the directories specified by the VCSDir directive.

-# `-#code`

Use the -# option to display diagnostic information for debugging purposes. Use the *code* parameter to select one of the following diagnostic functions:

`1`	Display each line of the configuration file as it is processed.
`2`	Display internal errors.
`4`	Toggle the sign-on message.
`500`	Display the archive owner, access list, access control database name, current user ID, and their privileges. The currently active privileges are in uppercase letters.

To specify more than one diagnostic item, add the numbers together. For example, use `-#3` to display the diagnostic information represented by numbers `1` and `2`.

The -# option should be the first option after the command. Version Manager processes commands in left-to-right order and will not display debugging information until it reaches the -# option.

-h Displays help for command-line options. The program terminates after it processes this option even if you specify other options.

-id **-id***user_id*:*user_password*

(File server only.) Specifies a user ID and password to authenticate against the access control database and/or LDAP server, if any, that is associated with the relevant file server path map in the Version Manager File Server Administration utility.

NOTE The `-id` option does not affect the user ID used to perform the CLI operation itself. The user ID for the CLI operation is determined in the normal manner.

-xe `-xefile_name`

Redirects status, program, and error messages to *file_name*.

-xo `-xofile_name`

Redirects standard output to *file_name*.

-xo+e `-xo+efile_name`

Redirects standard output and error messages to *file_name*.

Example The following commands make the files NEWFILE.TXT and CLONE.TXT identical:

```
vdiff -d oldfile.txt newfile.txt > file.dlt
    regen oldfile.txt file.dlt > clone.txt
```

Special Considerations

- For the REGEN program alone, you are granted an exception to the standard license agreement. You can make copies of REGEN.EXE and install them on an unlimited number of computers without regard to the usual "simultaneous use" restrictions. This also applies to the use of REGEN.C.
- By default, VDIFF and REGEN display their output. With either command, use either the redirection symbol (>) or the `-xo` option to redirect the output to a file.

Related Topics

For information about...	See...
Generating a delta file	"VDIFF command" on page 67
Other Version Manager commands	"Using Commands" on page 8

RSE command

Redirect error messages Use the RSE command to redirect another command's error messages from the standard error device to the standard output device. When commands are called from a batch file or a Configuration Builder build script, you can use RSE to prevent error messages from scrolling off the screen before you see them.

To use RSE with a single command, preface it with the RSE command. To use RSE with batch files, you have two options: preface each command in the batch file with the RSE command, or redirect output for the entire batch file.

Exit code RSE returns the exit code of the executed command, or 127 if the command could not be found.

Syntax `rse command [> output_file]`

Examples

- The following example redirects error messages generated by Configuration Builder to a file named BUILD.LOG:
  ```
  rse build > build.log
  ```
- The following example redirects error messages from commands called by the batch file DOIT.BAT to a file named DOIT.OUT:
  ```
  rse doit.bat > doit.out
  ```

Special Consideration **For UNIX users**: The RSE command is not distributed with the UNIX version of Version Manager.

Related Topics

For information about...	See...
Other Version Manager commands	"Using Commands" on page 8

VCOMPRES command

Compress or uncompress stored archive records

Use the VCOMPRES command to compress or uncompress information stored in existing archives. You can compress or uncompress change information (the *deltas*), the copy of the workfile stored in the archive (the *work image*), or both.

For new archives, Version Manager determines whether to compress data by reading the directives in effect at the time. If you use the VCS command to turn compression on or off after revisions have been stored in archives, it only affects revisions stored from then on.

Exit code

VCOMPRES returns an exit code of 0 if successful, 1 otherwise.

Syntax

`vcompres [option...] file_name...`

The following table lists frequently used commands. See the "Options," section below for details on all command-line options.

To do this...	Use this command...
Compress the work image and deltas	VCOMPRES
Compress the deltas only	VCOMPRES -D
Uncompress data	VCOMPRES -U
Compress the work image only	VCOMPRES -W

Options

@

@[list_file]

Use the `@` option to read *`list_file`* for additional command-line options. Version Manager reads additional options from the file before reading the rest of the command line. Use the `@` option with no parameter if you're redirecting input from another command.

If the file has the extension .GRP, you can omit the extension and the path. Version Manager searches for it in the directories specified by the VCSDir directive.

-#

`-#code`

Use the -# option to display diagnostic information for debugging purposes. Use the *`code`* parameter to select one of the following diagnostic functions:

`1`	Display each line of the configuration file as it is processed.
`2`	Display internal errors.
`4`	Toggle the sign-on message.

`500` Display the archive owner, access list, access control database name, current user ID, and their privileges. The currently active privileges are in uppercase letters.

To specify more than one diagnostic item, add the numbers together. For example, use `-#3` to display the diagnostic information represented by numbers `1` and `2`.

The `-#` option should be the first option after the command. Version Manager processes commands from left to right and does not display debugging information until it reaches the `-#` option.

-c `-c[- | configuration_file | directory_name]`

Instructs Version Manager to look for configuration information in a place other than in the master configuration file or the default configuration file.

-d Compresses deltas only. To uncompress deltas, use this option in conjunction with the `-u` option.

-h Displays help for command-line options. The program terminates after it processes this option even if you specify other options.

-id **`-id`***`user_id`*`:`*`user_password`*

(File server only.) Specifies a user ID and password to authenticate against the access control database and/or LDAP server, if any, that is associated with the relevant file server path map in the Version Manager File Server Administration utility.

NOTE The `-id` option does not affect the user ID used to perform the CLI operation itself. The user ID for the CLI operation is determined in the normal manner.

-q Selects the quiet mode of operation, in which Version Manager only reports error conditions.

-u Uncompresses data. Use this option with the `-d` option to uncompress deltas only or with the `-w` option to uncompress the work image only.

-w Compresses only the work image. To uncompress the work image, use this option in conjunction with the `-u` option.

-xe `-xe`*`file_name`*

Redirects status, program, and error messages to *`file_name`*.

-xo `-xo`*`file_name`*

Redirects standard output to *`file_name`*.

-xo+e `-xo+`*`efile_name`*

Redirects standard output and error messages to *`file_name`*.

Examples

- The following example uses the VCOMPRES command-line options in the list file COMPRESS.LST:
  `vcompres @compress.lst`
- The following example compresses the deltas in the archive HEADER.C_V:
  `vcompres -d header.c_v`
- The following example uncompresses the work image in the archive HEADER.C_V:
  `vcompres -u -w header.c_v`

- The following example compresses the workfile image in the archives MAIN.C_V and SCREEN.C_V:
  `vcompres -w main.c_v screen.c_v`

Special Consideration

The `-d` and `-w` options are mutually exclusive.

Related Topics

For information about...	See...
Enabling compression for existing archives	"VCS command" on page 56
Other Version Manager commands	"Using Commands" on page 8

VCONFIG command

Configure program files

Use the VCONFIG command to embed the master configuration file name, access control database name, and the list of user identification sources in the Version Manager VCONFIG files. A copy of the VCONFIG file must reside in the directory where Version Manager is installed.

Use this file...	In this environment...
VMWFVC.DLL	Windows
vmufvc.a	UNIX

Exit code

VCONFIG returns an exit code of 0 if successful, 1 otherwise.

Syntax

`vconfig [option...] file_name...`

Specify the files to configure or convert using the *file_name* parameter. File specifications can contain wildcards.

The following table lists frequently used commands. See the "Options," section below for details on all command-line options.

To do this...	Use this command...
Embed the access control database name	vconfig -a
Embed the master configuration file name	vconfig -c
Embed the user identification source	vconfig -i
Remove all embedded information	vconfig -u

If you don't specify any options, VCONFIG displays the current configuration of specified program files or archives.

Options

@ `@[list_file]`

Use the `@` option to read *`list_file`* for additional command-line options. Version Manager reads additional options from the file before reading the rest of the command line. Use the `@` option with no parameter if you're redirecting input from another command.

If the file has the extension .GRP, you can omit the extension and the path. Version Manager searches for it in the directories specified by the VCSDir directive.

-# `-#code`

Use the -# option to display diagnostic information for debugging purposes. Use the *`code`* parameter to select one of the following diagnostic functions:

`1`	Display each line of the configuration file as it is processed.
`2`	Display internal errors.
`4`	Toggle the sign-on message.
`500`	Display the archive owner, access list, access control database name, current user ID, and their privileges. The currently active privileges are in uppercase letters.

To specify more than one diagnostic item, add the numbers together. For example, use -#`3` to display the diagnostic information represented by numbers `1` and `2`.

The -# option should be the first option after the command. Version Manager processes commands from left to right and does not display debugging information until it reaches the -# option.

-a `-a[`*`database_name`*`]`

Embeds the name of the access control database in specified program files. To remove the currently embedded access control database name, omit the *`database_name`* parameter.

-c `-c[`*`file_name`*`]`

Embeds the name of the master configuration file in specified program files. If you don't supply the *`file_name`* parameter, Version Manager commands do not read the master configuration file.

-h Displays help for command-line options. The program terminates after it processes this option even if you specify other options.

-i `-i[`*`source`*`[,`*`source`*`...]]`

Embeds user identification sources and the order in which Version Manager uses them in specified program files. Version Manager commands check the sources in order from left to right and use the first one that produces a user ID.

Sources are not case-sensitive. If you specify more than one source, separate them with commas. Values for *source* are as follows:

Value	Meaning
Host ID	Host operating system. Use this source with systems that provide a user identification mechanism, such as UNIX, or for environments in which more than one network is in use. (Version Manager obtains the user ID from the operating system.) Use this option on UNIX and Windows systems where the user is required to enter a password. The directive for this option is LogIn=HOST.
LDAP ID	Lightweight Directory Access Protocol (LDAP). Use this source to authenticate user IDs and passwords against an LDAP server. Once authenticated against LDAP, user IDs are passed to the access control database, if one is in effect. The passwords, if any, in the access control database are ignored. The directive for this option is LogIn=LDAP. LDAP does not work with the command-line interface (CLI). If LDAP is the first login source specified, the CLI will attempt to use the next login source. If no other login sources are specified, the CLI command will fail.
Login Dialog	The Version Manager desktop client login utility. This source requires users to enter a password before they can use Version Manager. To use password protection, an access control database must be defined. This source is only available in the desktop client. The directive for this option is LogIn=VLOGIN.
Netware ID	Novell NetWare (Windows only). Use this source to obtain user IDs from a Novell NetWare server, rather than Windows. The directive for this option is LogIn=NETWARE.
VCS ID	The user's Version Manager ID, which Version Manager derives from the value of the VCSID environment variable. The directive for this option is LogIn=VCSID. Be aware that using VCSID as a source for user identification is not secure. Users can circumvent security by logging in as another user or resetting the value of the VCSID environment variable.
Wnet ID	Microsoft Windows networks. Version Manager obtains the user ID from the Microsoft WNET API. The directive for this option is LogIn=WNET.

Defaults The default for UNIX and Windows is HOST.

-id **-id***user_id*:*user_password*

(File server only.) Specifies a user ID and password to authenticate against the access control database and/or LDAP server, if any, that is associated with the relevant file server path map in the Version Manager File Server Administration utility.

NOTE The `-id` option does not affect the user ID used to perform the CLI operation itself. The user ID for the CLI operation is determined in the normal manner.

-l Specifies internationalization language.

-pl or -pm Specifies the locale or message path in the VCONFIG file. Use of this option typically relates to internationalization language.

-q Selects the quiet mode of operation where VCONFIG reports only error conditions.

-u Removes all information embedded in Version Manager program files, such as the name of the master configuration file, the name of the access control database, and the user ID sources.

-xe `-xe`*`file_name`*

Redirects status, program, and error messages to *`file_name`*.

-xo `-xo`*`file_name`*

Redirects standard output to *`file_name`*.

-xo+e `-xo+e`*`file_name`*

Redirects standard output and error messages to *`file_name`*.

Examples

- The following example displays the current configuration of all Version Manager program files in the current directory:

DOS
```
vconfig *.dll
```

UNIX
```
vconfig *
```

- The following example performs the command-line options contained in the file INSTALL.CFG on Version Manager program files in the current directory:

DOS
```
vconfig @install.cfg *.dll
```

UNIX
```
vconfig @install.cfg *
```

- The following example embeds the name and location of the access control database file C:\SERENA\ACCESS.DB for Windows and */usr/serena/access.db* for UNIX in Version Manager program files in the current directory:

DOS
```
vconfig -ac:\serena\access.db *.dll
```

UNIX
```
vconfig -a/usr/serena/access.db *
```

- The following example embeds C:\SERENA\MASTER.CFG for Windows and */usr/serena/master.cfg* for UNIX as the master configuration file name in all Version Manager program files in the current directory:

DOS
```
vconfig -cc:\serena\master.cfg *.dll
```

UNIX
```
vconfig -c/usr/serena/master.cfg *
```

- The following example embeds the user ID sources in Version Manager executables in */usr/serena*: the UNIX user ID, the value of the VCSID directive.

UNIX
```
vconfig -ihost,vcsid /usr/serena/*
```

- The following example removes embedded information from all Version Manager program files in the current directory:

DOS
```
vconfig -u *.dll
```

UNIX
```
vconfig -u *
```

Special Considerations

- Although the VCONFIG command is most often used during initial system configuration, you are free to modify the VCONFIG files at any time. Note that changes made using VCONFIG will not take effect on applications that are already running. The application must be restarted before the change will take effect.

- While configuring archives or program files, be sure that no one tries to access the files.
- Do not modify Version Manager program files except by means of VCONFIG. In particular, do not use any utility that converts your executables into self-extracting archives to save disk space.
- The access control database name defined with the AccessDB directive overrides the value embedded by VCONFIG -A.
- The VCONFIG command embeds only the name and location of a master configuration file in Version Manager program files, not the contents. Therefore, the master configuration file itself must always be available when you use Version Manager commands.
- If you use the wildcard character (*) to configure all files in the current directory, Version Manager issues an error message for any files that are not the VCONFIG file. You can ignore these error messages.
- For security reasons, this command is not placed in the same directory as other commands when Version Manager is installed. By default, it is installed in `\Bin\Admin` subdirectory of your Version Manager installation. If you require access to this command and do not have it, see your Version Manager administrator.
- It is important to control user's access to the VCONFIG command. Anyone who can use this command can change the names of the master configuration file or the access control database embedded in the Version Manager executables.
- **For UNIX users**: The VCONFIG -i command supports the source values HOST, LDAP, VCSID, and VLOGIN. Note that LDAP and VLOGIN do not support CLI commands.

Related Topics

For information about...	See...
Other Version Manager commands	"Using Commands" on page 8

VCS command

Perform administrative functions

Use the VCS command to create empty archives or to change archive attributes. For example, you can use the VCS command to change user access, assign version labels, and lock or unlock revisions.

Exit code

VCS returns an exit code of 0 if successful, 1 otherwise.

Syntax

`vcs [option...] file_name...`

The following table lists frequently used commands. See the "Options," section for details on all command-line options.

To do this...	Use this command...
Create a new archive without checking in a revision	vcs -i
Lock a revision	vcs -l

To do this...	Use this command...
Unlock a revision	vcs -u
Assign a version label	vcs -v

Default versions

If the DefaultVersion directive is not set and you don't specify a revision in a command, Version Manager operates on the tip revision of the trunk by default. The following commands are affected by how this directive is set:

```
vcs -l
vcs -r
vcs -v
```

When using the above commands, you can override the DefaultVersion directive by using a hyphen (-) after these command-line options.

Options

@ `@[list_file]`

Use the `@` option to read *`list_file`* for additional command-line options. Version Manager reads additional options from the file before reading the rest of the command line. Use the `@` option with no parameter if you're redirecting input from another command.

If the file has the extension .GRP, you can omit the extension and the path. Version Manager searches for it in the directories specified by the VCSDir directive.

-# `-#code`

Use the -# option to display diagnostic information for debugging purposes. Use the *`code`* parameter to select one of the following diagnostic functions:

`1`	Display each line of the configuration file as it is processed.
`2`	Display internal errors.
`4`	Toggle the sign-on message.
`500`	Display the archive owner, access list, access control database name, current user ID, and their privileges. The currently active privileges are in uppercase letters.

To specify more than one diagnostic item, add the numbers together. For example, use -#`3` to display the diagnostic information represented by numbers `1` and `2`.

The -# option should be the first option after the command. Version Manager processes commands from left to right and does not display debugging information until it reaches the -# option.

-a- `a[user_id[,user_id...]]`

Adds the names specified by the *`user_id`* parameter to the access list for specified archives. If you don't supply the *`user_id`* parameter, VCS removes all names from the access lists.

This option overrides the AccessList directive.

Privileges required. The -a option requires the ChangeAccessList privilege when used to modify existing archives.

-c `-c[- | configuration_file | directory_name]`

Instructs Version Manager to look for configuration information in a place other than in the master configuration file or the default configuration file.

-ec `-ecstring`

Changes the comment prefix used when expanding the Log keyword to *string*. If the *string* parameter contains spaces, enclose it in single or double quotation marks (' or ").

You can place any character in the *string* parameter using the form \0X*nn*, where *nn* is the hexadecimal equivalent of the desired character. Version Manager interprets the character \0X00 as the end of the string.

Privileges required. The -ec option requires the ChangeCommentDelimiter privilege when used to modify existing archives.

-en `-enstring`

Changes the newline string used when expanding the Log keyword to *string*. If the *string* parameter contains spaces, enclose it in single or double quotation marks (' or ").

Use the following symbols to indicate special characters in the *string* parameter:

\r	Carriage return
\n	Line feed
\t	Tab

The default is "\r\n".

You can place any character in the *string* parameter using the form \0Xnn, where *nn* is the hexadecimal equivalent of the desired character. Version Manager interprets the character \0X00 as the end of the string.

Privileges required. The -en option requires the ChangeCommentDelimiter privilege when used to modify existing archives.

-g `-gpromotion_group:[revision_number |:version_label]`

Associates a revision with a promotion group. Version Manager associates the promotion group with the revision number or version label specified by the *:revision* or *:version_label* parameter.

Privileges required. The -g option requires the AddGroup privilege to assign a promotion group to a revision the first time. The DeleteGroup and ModifyGroup privileges are required to rename a promotion group. The ModifyGroup privilege is required to reassign a promotion group to a different revision.
`-gpromotion_group:delete`

Use the *:delete* parameter to remove the association with the specified promotion group. Enter an asterisk (*) for the promotion_group parameter to remove all promotion groups.

Privileges required. The DeleteGroup privilege is required to delete a promotion group.

-h Displays help for command-line options. The program terminates after it processes this option even if you specify other options.

-i Creates an archive that contains no revisions. If the archive already exists, the program prompts you for permission to overwrite it.

Use the *file_name* parameter (shown in the syntax statement for the VCS command) to specify the name of the workfile.

When you use the -i option, the program ignores the following options: -l, -m, -r, -u, -v, and -w.

Privileges required. The -i option requires the InitArchive privilege.

-id **-id***user_id*:*user_password*

(File server only.) Specifies a user ID and password to authenticate against the access control database and/or LDAP server, if any, that is associated with the relevant file server path map in the Version Manager File Server Administration utility.

NOTE The -id option does not affect the user ID used to perform the CLI operation itself. The user ID for the CLI operation is determined in the normal manner.

-l -l[*revision_number* | *version_label* | *promotion_group*]

Locks the specified revision. You can specify the revision or the version label as an additional parameter, or use the -r or -v option.

When you use this option, Version Manager displays the revision number that is reserved for this revision when you check it in. No other user can use this revision number in this archive until the promised check in number is released.

Privileges required. This option requires the Lock, LockTip, or LockNonTip privileges.

-m Specifies the description of the changes to the revision. If you don't specify a parameter to this option, the program prompts you to enter the description.

The program specifies the description for the trunk's tip revision unless you use the -r or -v option to specify a revision.
-m*text*

Use the -m option with a *text* parameter to specify the change description on the command line. If *text* contains spaces, enclose it in double quotation marks ("). If *text* begins with an at sign (@), precede it with a backslash (\).
-m@*file_name*

Use the -m option with the *@file_name* parameter to read the change description from *file_name*.
-m@

Use the -m option with the @ parameter to read the change description from the file whose name is computed from the workfile name using the value specified by the MessageSuffix directive.
-m@*directory*

Use the -m option with the *@directory* parameter to read the change description from the file located in *directory* whose name is derived from the workfile.

Privileges required. The VCS -m option requires the ModifyDescription or ModifyChangeDescription privilege.

-n Answers no in advance to any questions VCS may ask you. The possible questions are:

- Should VCS overwrite an existing file when you use the `-i` option to create an archive?
- Should VCS delete a previous version label that is identical to the new version label?
- Should VCS reuse the previous archive description or change description?

See also the `-q` and `-y` options.

-o `-ouser_id`

Changes the owner of the specified archives to *`user_id`*.

Privileges required. The `-o` option requires the ChangeOwner privilege when used to modify existing archives.

pc `[-|+]pc`

Enables (+) or disables (-) compression of workfile images. For details, see the "CtrlZ directive" on page 133. (This affects all future deltas, but not existing deltas.)

pd `[-|+]pd`

Enables (+) or disables (-) compression of deltas. For details, see the "CompressDelta directive" on page 132. (This affects all future deltas, but not existing deltas.)

pe `[-|+]pe`

Enables (+) or disables (-) exclusive locking. For details, see the "ExclusiveLock directive" on page 148. (This affects all future deltas, but not existing deltas.)

pg `[-|+]pg`

Enables (+) or disables (-) delta generation for an archive. (This affects all future deltas, but not existing deltas.)

pk `[-|+]pk`

Enables (+) or disables (-) keyword expansion. For details, see the "ExpandKeywords directive" on page 149. (This affects all future deltas, but not existing deltas.)

pl `[-|+]pl`

Enables (+) or disables (-) lock checking. For details, see the "CheckLock directive" on page 128. (This affects all future deltas, but not existing deltas.)

pt `[-|+]pt`

Enables (+) or disables (-) file translation. For details, see the "Translate directive" on page 174. (This affects all future deltas, but not existing deltas.)

pw `[-|+]pw`

Enables (+) or disables (-) write protection of archives. For details, see the "WriteProtect directive" on page 179. (This affects all future deltas, but not existing deltas.)

Privileges required. The `pc`, `pd`, `pg`, `pe`, `pk`, `pl`, and `pw` options require the ChangeProtection privilege.

-q Selects the quiet mode of operation, in which Version Manager only reports error conditions. The `-q` option answers yes in advance to any questions VCS may ask.

-qo Selects the quiet mode of operation without giving the program permission to overwrite existing archives. When `-qo` is used with the `-y` option, it is equivalent to `-q`.

-qn Answers no in advance to any question VCS may ask you. This option has the same effect as using the `-n` option with either the `-q` or `-qo` option. See also `-n`.

-r `-r[revision_number]`

Specifies the revision number you want to modify. If you use `-r` without a revision number, the program uses the latest trunk revision, unless DefaultVersion is defined.

-s `-ssuffix_template`

Specifies the archive suffix template to use instead of the current value of ArchiveSuffix.

-t Specifies the workfile description. See also `-m`.

If you don't specify a parameter for this option, the program prompts you to enter the description.
`-ttext`

Use the `-t` option with a *`text`* parameter to specify the workfile description on the command line. If `text` contains spaces, enclose it in double quotation marks ("). If `text` begins with an at sign (`@`), precede it with a backslash ().
`-t@file_name`

Use the `-t` option with the *`@file_name`* parameter to read the workfile description from *`file_name`*.
`-t@`

Use the `-t` option with the `@` parameter to read the workfile description from the file whose name is computed from the workfile name using the value specified by the MessageSuffix directive.
`-t@directory`

Use the `-t` option with the `@directory` parameter to read the workfile description from the file located in *`directory`* whose name is derived from the workfile.

Privileges required. The `-t` options require the ModifyDescription or ModifyWorkfileDescription privilege when modifying existing archives.

-u `-u[revision_number | version_label | promotion_group]`

Unlocks the specified revision. You can specify the revision as a parameter to this option, or with the `-r` or `-v` option. If you specify the revision by branch number only (such as 1.2.1), VCS unlocks the latest revision on the branch.
`-u:[user | *]`

Use the `-u` option with a colon plus the *`user`* parameter to remove all locks by that user. Enter an asterisk (*) for the *`user`* parameter to remove all locks by all users in the archive.

Privileges required. The `-u` option requires the Unlock privilege to remove your lock. It requires the BreakLock privilege to remove a lock placed by another user.

-v `-vversion_label`

Use this form of the `-v` option without any other command-line options to assign *`version_label`* to the tip revision of the trunk. Use this form of the `-v` option with other command-line options to identify a revision to be operated on.

If *version_label* begins with a digit, precede it with a backslash.

If the DefaultVersion directive is set, the above command assigns the version label to a place other than the trunk tip revision.

-vversion_label:[*revision_number* | *version_label* | *promotion group*]

Use this form of the -v option, or use the -v option plus the -r option, to apply the version label to the specified revision number, version label, or promotion group. If *version_label* is already assigned to another revision, the program prompts you for permission to remove it. You can use either the -q or -y option to provide permission, or the -n option to deny permission.

Privileges required. This option requires the AddVersion privilege if the version label is new, or the ModifyVersion and DeleteVersion privileges if the version label already exists.

-vnew_version_label::*old_version_label*

Use this form of the -v option to replace *old_version_label* with *new_version_label*.

Privileges required. This option requires the AddVersion privilege if the version label is new, or the ModifyVersion and DeleteVersion privileges if the version label exists.

-vversion_label:delete

Use this form of the -v option to delete *version_label* from specified archives.

Privileges required. This option requires the DeleteVersion privilege.

-vfloating_label:*revision_number**

Use this form of the -v option to associate *floating_label* with the revision associated with *revision_number* in specified archives. Floating labels "float" with the tip revision of the branch or trunk to which they are assigned.

-vfloating_label:*version_label**

Use this form of the -v option to associate *floating_label* with the revision associated with *version_label* in specified archives.

Privileges required. This option requires the AddVersion privilege if the version label is new, or the ModifyVersion and DeleteVersion privileges if the version label exists.

-vversion_label:*

Use this form of the -v option to transform *version_label* into a floating label on the trunk or the branch specified by DefaultVersion in specified archives.

Privileges required. This option requires the AddVersion privilege if the version label is new, or the ModifyVersion and DeleteVersion privileges if the version label exists.

-w *-wworkfile_name archive_name*

Changes the workfile name stored in the archive. This option does not change the name of the archive itself. The specified *archive_name* must include the archive's extension. If you rename the archive, use this option to change the name of the workfile correspondingly.

Privileges required. The -w option requires the ChangeWorkfileName privilege.

-xcolumnmask *-xcolumnmask="start-end[(numeric)]..."*

Use this option to change the columns that Version Manager converts to spaces when generating change information. This option overrides the ColumnMask directive.

The *start* and *end* parameters specify column numbers. Column numbering begins with column number 1. You can define more than one column range on the command line, and you can specify them in any order. Ranges can overlap. To turn off column masking for the archive, use zeroes for the column range.

To restrict masking to numeric fields only, use the keyword *(numeric)* following a column range. In this case, the program only masks a field if the first character of the field is numeric.

```
-xcolumnmask=cobol
```

This form of the `-xcolumnmask` option is equivalent to the following:

```
-xcolumnmask="1-6(numeric)" "73-80"
```

-xe `-xefile_name`

Redirects status, program, and error messages to *file_name*.

-xo `-xofile_name`

Redirects standard output to *file_name*.

-xo+e `-xo+efile_name`

Redirects standard output and error messages to *file_name*.

-xrecordlength `-xrecordlength =record_length`

Changes the record length of the workfile. The *record_length* parameter specifies the number of characters. The maximum record length is 64K. To turn off the record length, use zero.

This option overrides the Renumber directive.

-xrenumber `-xrenumber =start-end [from start by number]`

Specifies the column range that the program renumbers when retrieving the workfile. The *column_start* and *column_end* parameters specify column numbers. Column numbering begins with column number 1. *start* is the number to begin numbering with and *number* is the amount by which to increase this number for each line. Version Manager fills line numbers with zeroes beginning on the left (000010, 000020, etc.). Using -xrenumber=0 turns off column renumbering.

For COBOL files, use the following predefined value:

```
-xrenumber =cobol
```

This form of the `-xrenumber` option is equivalent to the following:

```
Renumber 1-6 from 10 by 10
```

-y Answers yes in advance to any question the program may ask you. The effect is similar to that of the `-q` option, except that `-q` also suppresses some informative output.

Examples

- The following example locks the trunk tip revision of the archive associated with TEST.PAS:

  ```
  vcs -l test.pas
  ```

- The following example enables lock checking for the archive TEST.PAV:

  ```
  vcs +pl test.pav
  ```

- The following example creates an archive and disables keyword expansion for the workfile TEST.EXE for Windows and *test* for UNIX:

DOS
```
vcs -i -pk test.exe
```

UNIX
```
vcs -i -pk test
```

- The following example changes the newline string used in Log keyword expansion to space, asterisk, slash, carriage return, line feed.
```
vcs -en" */\r\n" *.c_v
```

Special Considerations

- When creating archives with the VCS command, you must name the workfile explicitly.
- Before you use the `-l`, `-r`, or `-v` options, make sure that if the DefaultVersion directive is set, it relates to a revision you want to work on. You may want to tell Version Manager to ignore the directive's setting by using a hyphen after the option, or redefine the directive in your configuration file. Otherwise, VCS will work on a revision other than the one intended.

Related Topics

For information about...	See...
Specifying an editor	"VCSEdit directive" on page 177
Archive extensions	"ArchiveSuffix directive" on page 122
Defining a default revision or version	"DefaultVersion directive" on page 135
Specifying record length	"RecordLength directive" on page 165
Renumbering lines	"Renumber directive" on page 167
Other Version Manager commands	"Using Commands" on page 8
Keywords	*Administrator's Guide*

VDEL command

Delete one or more revisions from archives

Use the VDEL command to delete revisions from archives. The changes reflected in these revisions are placed in the first revision following the revision or range of revisions deleted. Changes are lost only if the tip revision is deleted.

When a revision is deleted, corresponding change descriptions are also deleted, unless they were embedded in the workfile using the Log keyword.

Privileges required. This command requires the DeleteRev, DeleteRevNonTip, or DeleteRevTip privilege.

Exit code

VDEL returns an exit code of 0 if successful, 1 otherwise.

Syntax

```
vdel [option...] file_name...
```

The following table lists frequently used commands. See "Options," below, for details on all command-line options.

To do this...	Use this command...
Delete revisions	vdel -r
Delete revisions identified by version label	vdel -v

Default versions

The DefaultVersion directive affects the following commands:

vdel -r
vdel -v

If the DefaultVersion directive is set, you can override it by using a hyphen (-) after the above options.

Options

@ `@[list_file]`

Use the `@` option to read *`list_file`* for additional command-line options. Version Manager reads additional options from the file before reading the rest of the command line. Use the `@` option with no parameter if redirecting input from another command.

If the file has the extension .GRP, you can omit the extension and the path. Version Manager searches for it in the directories specified by the VCSDir directive.

-# `-#code`

Use the -# option to display diagnostic information for debugging purposes. Use the *`code`* parameter to select one of the following diagnostic functions:

`1`	Display each line of the configuration file as it is processed.
`2`	Display internal errors.
`4`	Toggle the sign-on message.
`500`	Display the archive owner, access list, access control database name, current user ID, and their privileges. The currently active privileges are in uppercase letters.

To specify more than one diagnostic item, add the numbers together. For example, use -#`3` to display the diagnostic information represented by numbers `1` and `2`.

The -# option should be the first option after the command. Version Manager processes commands from left to right and does not display debugging information until it reaches the -# option.

-c `-c[- | configuration_file | directory_name]`

Instructs Version Manager to look for configuration information in a place other than in the master configuration file or the default configuration file.

See "Configuration Files" on page 111 for details on using the `-c` option.

-h Displays help for command-line options. The program terminates after it processes this option even if you specify other options.

-id **-id***user_id*:*user_password*

(File server only.) Specifies a user ID and password to authenticate against the access control database and/or LDAP server, if any, that is associated with the relevant file server path map in the Version Manager File Server Administration utility.

NOTE The -id option does not affect the user ID used to perform the CLI operation itself. The user ID for the CLI operation is determined in the normal manner.

-q Selects the quiet mode of operation, in which Version Manager only reports error conditions. The -q option gives permission in advance to delete revisions.

-qo Direct the program to report only error codes, and to ask permission before deleting revisions. When you use -qo with the -y option, it is equivalent to -q.

-r -r[*revision_number* | *range*][+]

Specifies the revision number or range of revisions to be deleted. The range is specified using any combination of revision numbers. If you use -r without a revision number, the program uses the latest trunk revision, unless DefaultVersion is defined. If the version label begins with a number, precede it with a backslash (\).

To delete all branches from a revision or range of revisions, follow the version label or range with a plus sign (+).

-s -s*suffix_template*

Specifies the archive suffix template to use instead of the current value of ArchiveSuffix.

-v -v*version_label* | *version_range*[+]

Specifies revisions to be deleted using version labels. To delete all revisions between two version labels, enter a range of version labels.

To delete all branches from a revision or range of revisions, follow the version label or range with a plus sign (+).

To delete the first revision on a branch or the trunk that is associated with a version label, use *-vversion_label +#*.

To delete the last revision on a branch or the trunk that is associated with a version label, use *-vversion_label -#*.

-xe -xe*file_name*

Redirects status, program, and error messages to *file_name*.

-xo -xo*file_name*

Redirects standard output to *file_name*.

-xo+e -xo+e*file_name*

Redirect standard output and error messages to *file_name*.

-y Gives permission in advance to delete revisions. The effect is similar to that of the -q option, except that -q also suppresses some informative output.

Examples

- The following example deletes Revision 1.2:
  ```
  vdel -r1.2 prog.c
  ```
- The following example deletes revision 1.2 and all branches that originate from it:

```
vdel -r1.2+ prog.c
```

- The following example deletes branch 1.2.5:
  ```
  vdel -r1.2.5 prog.c
  ```
- The following example deletes the revisions between and including 1.2 and the tip revision:
  ```
  vdel -r1.2* prog.c
  ```
- The following example deletes revisions 1.0 through 1.2:
  ```
  vdel -r*1.2 prog.c
  ```
- The following example deletes branch revisions 1.3.1.5 through 1.3.1.8 and all branches that originate from these revisions:
  ```
  vdel -r1.3.1.5*1.3.1.8+ prog.c
  ```
- The following example deletes all revisions between and including the version label *alpha* and the version label *beta*:
  ```
  vdel -valpha*beta prog.c
  ```
- The following example deletes all revisions between and including the version label *alpha* and the version label *beta*, including any branches that originate from those revisions:
  ```
  vdel -valpha*beta+ prog.c
  ```

Special Considerations

- The VDEL command removes revisions from an archive; it does not undo the changes those revisions represent. To undo the changes made in one or more revisions, use the VMRG command.
- You cannot delete a locked revision.
- You cannot delete the revision from which a branch originates unless you use a plus sign (+) to delete the branch as well.

Related Topics

For information about...	See...
Using the VMRG command to undo changes in a revision	"VMRG command" on page 81
Defining a default revision or version	"DefaultVersion directive" on page 135
Other Version Manager commands	"Using Commands" on page 8
Keywords	*Administrator's Guide*

VDIFF command

Display the differences between two revisions

Use the VDIFF command to compare two files and report their differences. VDIFF displays differences in terms of how the first file (the *reference file*) was modified to produce the second file (the *target file*). You can use VDIFF to compare the following types of files:

- Text file
- Checked-out workfile
- Revision in an archive

You can also use VDIFF to generate a delta file, which contains editing instructions which the REGEN command can use to regenerate the target file from the reference file.

The section "Sample VDIFF Output" on page 72 shows how VDIFF reports differences between files.

Privileges required. The VDIFF command requires the Get privilege to check out revisions from archives.

Exit code

VDIFF returns an exit code of 0 if successful, 1 otherwise. If you use the `-t` option, it returns an exit code of 0 if the files are identical, an exit code of 1 if a problem occurs, and an error code of 2 if the files are different.

Syntax

`vdiff [option...] reference_file target_file`

Specifying files

Use the following rules when specifying files to compare:

- To compare a workfile, specify its file name.
- To compare a revision in an archive, specify the workfile name preceded by a revision number or version label. Use the `-r` option to specify the revision number, or the `-v` option to specify the version label.
- To compare the latest trunk revision in the archive, specify the archive.
- To compare a file with the same name in a different directory, specify the directory name.
- You cannot use wildcards to specify files for the VDIFF command.
- You cannot use date ranges.

The following table lists frequently used commands. See the "Options" section below for details on all command-line options.

To do this...	Use this command...
Specify revision or version for comparison	vdiff -r
Specify a revision for comparison by version label	vdiff -v
Send VDIFF output to a file	vdiff -xo

Default versions

The DefaultVersion directive affects the following commands.

vdiff -r
vdiff -v

You can override this directive by using a hyphen (-) after the option. When you override the DefaultVersion directive, you compare the tip revision of the trunk.

Binary Files

VDIFF will perform a byte by byte comparison for a binary file. If there are no CRLFs in the file, then VDIFF compares in 512K-byte blocks.

Options

@

`@[list_file]`

Use the `@` option to read *`list_file`* for additional command-line options. Version Manager reads additional options from the file before reading the rest of the command

line. Use the @ option with no parameter if you're redirecting input from another command.

If the file has the extension .GRP, you can omit the extension and the path. Version Manager searches for it in the directories specified by the VCSDir directive.

-# -#*code*

Use the -# option to display diagnostic information for debugging purposes. Use the *code* parameter to select one of the following diagnostic functions:

1	Display each line of the configuration file as it is processed.
2	Display internal errors.
4	Toggle the sign-on message.
500	Display the archive owner, access list, access control database name, current user ID, and their privileges. The currently active privileges are in uppercase letters.

To specify more than one diagnostic item, add the numbers together. For example, use -#3 to display the diagnostic information represented by numbers 1 and 2.

The -# option should be the first option after the command. Version Manager processes commands from left to right and does not display debugging information until it reaches the -# option.

-a Displays all differences. By default, when VDIFF detects a block of lines that have been moved, it only displays difference information in the part of the file to which the lines were moved. Use the -a option to display difference information in the part of the file from which the lines were moved also.

-b Ignores white space at the beginning and end of the line during the comparison process (with this option, white space between words is not ignored). White space characters are spaces, tabs, carriage returns, line feeds, and form feeds.

-c -c[- | *configuration_file* | *directory_name*]

Instructs Version Manager to look for configuration information in a place other than in the master configuration file or the default configuration file.

See "Configuration Files" on page 111 for details on using the -c option.

-d Generates a delta file, which contains editing commands used by the REGEN command to generate the target file from the reference file.
-d@*delta_config_file*

Use the -d option with the @*delta_config_file* parameter to read the specified configuration file which contains delta configuration directives.

-dl Generates a delta file in CA-LIBRARIAN format. This option overrides any delta editing directives in the configuration file.

-dp Generates a delta file in CA-PANVALET format. This option overrides any delta editing directives in the configuration file.

-ds Generates a delta file in Version Manager format. This is equivalent to the -d option. This option overrides any delta editing directives in the configuration file.

-e -e[*number*]

Specifies the number of columns between tab stops for tab expansion. When VDIFF displays difference information, it replaces tab characters with spaces. The default number of columns per tab stop is four.

If you don't want VDIFF to expand tabs, use `-eo` or `-e`.

-h Displays help for command-line options. The program terminates after it processes this option even if you specify other options.

-id **-id***user_id*:*user_password*

(File server only.) Specifies a user ID and password to authenticate against the access control database and/or LDAP server, if any, that is associated with the relevant file server path map in the Version Manager File Server Administration utility.

NOTE The `-id` option does not affect the user ID used to perform the CLI operation itself. The user ID for the CLI operation is determined in the normal manner.

-l `-l`*number*

Specifies the number of context lines displayed before and after an area of change. The default is to display two lines of context.

This option also controls the size of change regions displayed, because VDIFF looks for *2number+1* unchanged lines following a changed line before it decides that a region of change has ended. If you don't specify *number*, VDIFF displays the entire file in the difference report.

-n Eliminates line numbers from the VDIFF report. VDIFF only displays the change type characters (+, -, < and >). The resulting report is then only one column wider than the compared text itself.

-q Selects the quiet mode of operation, in which Version Manager only reports error conditions.

-r `-r[`*revision_number*`]`

Specifies the revision to be compared. If you use the `-r` option without a revision number, the program compares the latest revision on the trunk, unless the DefaultVersion directive is defined.

-s *-ssuffix_template*

Specifies the archive suffix template to use instead of the current value of ArchiveSuffix.

-t Selects the test mode of operation. In test mode, VDIFF does not display difference information, but returns an exit code of 0 if the files are identical, an exit code of 1 if a problem occurs, and an error code of 2 if the files are different. The `-t` option has precedence over the `-d` option. With this option, VDIFF exits after the first difference is found

-v *-vversion_label*

Specifies the version label of the revision to be compared.

-xcolumnmask `-xcolumnmask = "`*start-end* `[(numeric)]..."`

Changes the columns that Version Manager converts to spaces when generating change information. This option overrides the ColumnMask directive.

The *start* and *end* parameters specify column numbers. Column numbering begins with column number 1. You can define more than one column range on the command line, and you can specify them in any order. Ranges can overlap. To turn off column masking for an archive, use zeroes for the column range.

To restrict masking to numeric fields only, use the keyword (numeric) following a column range. In this case, the program only masks a field if the first character of the field is numeric.

```
-xcolumnmask = cobol
```

This form of the `-xcolumnmask` option is equivalent to:

```
-xcolumnmask = "1-6 (numeric)" "73-80"
```

-xe `-xefile_name`

Redirects status, program, and error messages to *file_name*.

-xo `-xofile_name`

Redirects file difference output to *file_name*.

-xo+e `-xo+efile_name`

Redirects file difference output and status, program, and error messages to *file_name*.

-xrecordlength `-xrecordlength = record_length`

Changes the record length of the workfile. The *record_length* parameter is the number of characters. The maximum record length is 64K.

This option overrides the RecordLength directive.

Examples For this example, assume that you have two text files and two archives named TEST1.TMP, TEST2.TMP, TEST1.C_V, and TEST2.C_V, and that TEST1.C is checked out but TEST2.C is not.

- The following example compares a workfile with the tip revision of the archive:
  ```
  vdiff -r test1.c
  ```
- The following example compares a text file with the tip revision of the archive:
  ```
  vdiff test1.tmp -r test1.c
  ```
- The following example compares the two text files:
  ```
  vdiff test1.tmp test2.tmp
  ```
- The following example compares a text file with the specified revision in an archive:
  ```
  vdiff test1.tmp -r2.1 test1.c
  ```
- The following example compares a workfile with the tip revision in a different archive:
  ```
  vdiff test1.c -r test2.c
  ```
- The following example compares two revisions in the same archive:
  ```
  vdiff -r1.3 -r1.4 test1.c
  ```
- The following example compares two revisions in different archives:
  ```
  vdiff -r1.3 test1.c -r2.1 test2.c
  ```
- The following example compares the latest two revisions in the archive associated with TEST1.C:
  ```
  vdiff -r-1 -r test1.c
  ```
- The following example tests whether two files are the same:
  ```
  vdiff -t file1 file2
  ```

- The following example generates a delta file and redirects it to a file named DELTA:
  `vdiff -d file1 file2 -xodelta`

Sample VDIFF Output

Interpret the results of file comparison

The VDIFF report tells how the reference file was transformed into the target file. The names, revision numbers (if applicable), and timestamp of the two files compared appear at the top of the file. The report then shows change information for each region of difference. Change regions are delineated by strings of equals signs (=).

VDIFF compares files on the basis of lines of text. It treats a line in which a single character was changed as a line that was deleted and then inserted.

The sample report below shows the differences between a modified workfile and the latest checked in revision:

```
 TEST.C_V Rev 2.1 (10 Apr 1986  8:40:42)
              TEST.C (10 Apr 1986  9:46:24)
====================
      72      72  |
      73      73  |/** public variables **/
-     74          |int touch = FALSE;
-     75          |int writable = FALSE;
      76      74  |
      77      75  |/** local variables **/
====================
      87      85  |STATIC int islog1 = FALSE;
      88      86  |STATIC int islog2 = FALSE;
+             87  |STATIC int test = 0;
      89      88  |
      90      89  |/** public functions **/
====================
     327     321  |    }
     328     322  |
> (  317)    323  |    tloc = tlineno * sizeof(tl);
> (  318)    324  |    status = v_get(tid, sizeof(tl));
> (  319)    325  |    if (status == -1)
> (  320)    326  |      vf_error(tgtvfid, "temp ");
> (  321)    327  |
     329     328  |    /* check for matched line */
     330     329  |    if (tl.loc & LINE_MATCH)
====================
```

Each line of change information begins with a character that indicates the nature of the change, according to the table below.

This symbol...	Indicates this type of change...
+	A line is present in *target_file* but not in *reference_file*.
-	A line is present in *reference_file* but not in *target_file*.
>	A line was moved from some other location in *reference_file*.
<	A line was moved to some other location in *target_file*.

If none of these characters appear, the line is the same in both files.

The two columns of numbers show the line numbers associated with the line in each file. If lines were moved, the number in parentheses indicates the source or destination line number. When the line begins with a greater-than sign (>), the number in parentheses is the source line number; otherwise, the number in parentheses represents the destination. The vertical bar (|) indicates the beginning of the actual line.

Special Considerations

- You cannot use wildcards to specify files for VDIFF.
- By default, VDIFF displays its output. Use the `-xo` option to redirect output to a file.

Related Topics

For information about...	See...
Recreating a file from its delta file	"REGEN command" on page 47
Defining a default revision or version	"DefaultVersion directive" on page 135
Archive suffixes	"ArchiveSuffix directive" on page 122
Other Version Manager commands	"Using Commands" on page 8

VJOURNAL command

Display journal file information

Use the VJOURNAL command to view a report of changes to archives. Changes are recorded in *journal files*, which are text files that contain a record of each operation that changes an archive.

The VJOURNAL report reflects changes made to archives made using the GET, PUT, and VCS commands.

Privileges required. The VJOURNAL command requires the JournalReport privilege to view the change report.

Exit code

VJOURNAL returns an exit code of 0 if successful, 1 otherwise.

Syntax

`vjournal [option...] [journal_file...]`

The *journal_file* parameter can include wildcards. If you specify multiple files, VJOURNAL displays their contents in order.

If you don't specify a journal file, the program uses the file specified by the Journal directive. If this directive does not specify a file, VJOURNAL searches the directories named by the VCSDir directive and uses all of the journal files it finds to generate the report.

The following table lists frequently used commands. See the "Options," section that follows for details on all command-line options.

To view...	Use this command...
Changes made between certain dates	vjournal -d
Changes made to certain archives	vjournal -l
Changes made using certain commands	vjournal -o

To view...	Use this command...
Changes made by certain users	vjournal -u
Currently locked revisions	vjournal -xl

Options

@ `@[list_file]`

Use the `@` option to read *`list_file`* for additional command-line options. Version Manager reads additional options from the file before reading the rest of the command line. Use the `@` option with no parameter if you're redirecting input from another command.

If the file has the extension .GRP, you can omit the extension and the path. Version Manager searches for it in the directories specified by the VCSDir directive.

-# `-#code`

Use the -# option to display diagnostic information for debugging purposes. Use the *`code`* parameter to select one of the following diagnostic functions:

`1`	Display each line of the configuration file as it is processed.
`2`	Display internal errors.
`4`	Toggle the sign-on message.
`500`	Display the archive owner, access list, access control database name, current user ID, and their privileges. The currently active privileges are in uppercase letters.

To specify more than one diagnostic item, add the numbers together. For example, use -#`3` to display the diagnostic information represented by numbers `1` and `2`.

The -# option should be the first option after the command. Version Manager processes commands from left to right and does not display debugging information until it reaches the -# option.

-c `-c[- | configuration_file | directory_name]`

Instructs Version Manager to look for configuration information in a place other than in the master configuration file or the default configuration file.

See "Configuration Files" on page 111 for details on using the `-c` option.

-d `-ddate_range`

Limits the report to changes made within the specified range of dates.

-h Displays help for command-line options. The program terminates after it processes this option even if you specify other options.

-id **`-id`***`user_id:user_password`*

(File server only.) Specifies a user ID and password to authenticate against the access control database and/or LDAP server, if any, that is associated with the relevant file server path map in the Version Manager File Server Administration utility.

NOTE The `-id` option does not affect the user ID used to perform the CLI operation itself. The user ID for the CLI operation is determined in the normal manner.

-l `-larchive[,archive...]`

Limits the report to certain archives. Wildcards are allowed in archive names but not in path names. Separate multiple archive names with commas.

If you specify an archive without a path, for example, main.c_v, then all directories containing main.c_v are included in the report. If you specify a workfile, for example, main.c, then main.c is converted to an archive and reported as above.

-o `-ocommand[,command...]`

Limits the report to changes made using specific Version Manager commands. The *command* parameter can contain the following values: GET, PUT, VDEL, VPROMOTE, and VCS. For example, you can specify PUT -r and Version Manager will match however many characters you specify. Separate multiple commands with commas.

-s `-sdelimiter_character`

Specifies a delimiter character for use in parsing filenames. Use this when filenames may contain the default delimiter character of a comma (,). This is relevant when using the - L option.

-u `-uuser_id[,user_id...]`

Limits the report to changes made by certain users. Separate multiple user IDs with commas.

-xe `-xefile_name`

Redirects status, program, and error messages to *file_name*.

-xl `-xl[user_id[,user_id...]]`

Produces a report of locked revisions. Use the *user_id* parameter to see a report only of revisions locked by those users. Separate multiple user IDs with commas.

-xo `-xofile_name`

Redirects standard output to *file_name*.

-xo+e `-xo+efile_name`

Redirects standard output and error messages to *file_name*.

Examples

- The following example displays the journal entries that resulted from the PUT command:
  ```
  vjournal -oput
  ```
- The following example displays journal entries made by users aaronn, elvisc, and admin for all archives with the extensions .C_V and .PAV contained in journal files in the current working directory:
  ```
  vjournal -uaaronn,elvisc,admin -l*.c_v,*.pav *.jnl
  ```
- The following example displays journal entries made by user patsyc during January 1992 contained in the journal file PROJECT.JNL:
  ```
  vjournal -upatsyc -d"jan 92" project.jnl
  ```

- The following example creates a file named JOURNAL.RPT, which contains all journal entries for archives in the C:\PROJ directory for Windows and */proj* for UNIX contained in the journal file PROJECT.JNL:

DOS
```
vjournal -lc:\proj\*.??v -xojournal.rpt project.jnl
```

UNIX
```
vjournal -l/proj/*.??v -xojournal.rpt project.jnl
```

Special Considerations

- Unless you use the `-xo` option, the VJOURNAL command displays its report. If you use the VJOURNAL command without options, it displays the entire journal file.
- If you use multiple options that restrict the report, the resulting report shows the intersection of the options. For example, the following command shows changes made between 1/1/92 and 1/15/92 by user lauran.
  ```
  vjournal -d1/1/92*1/15/92 -ulauran
  ```
- The VLOG -L command produces the same report of locked revisions as the VJOURNAL -XL command. VJOURNAL is much faster, because it does not have to read each archive to obtain the information. However, the VJOURNAL report may be inaccurate or unavailable. Causes of inaccuracy include users with inconsistent configuration files, locked archives being moved, and the journal file being unavailable while revisions are being locked or unlocked.

Related Topics

For information about...	See...
Enabling journal entries	"Journal directive" on page 154
Specifying date and time ranges	"DateFormat directive" on page 134
Related commands	"VLOG command" on page 76
Other commands	"Using Commands" on page 8

VLOG command

Generate reports from archives

Use the VLOG command to view the following types of information about archives:

- Archive and workfile names
- Archive descriptions
- The user ID of the locker(s), if any
- Revision numbers
- Revision descriptions
- Revision history

Privileges required. The VLOG command requires the ViewArchive or ViewArchiveHeader privilege to view header information. It requires the ViewArchive or ViewArchiveRev privilege to view information about revision history.

Exit code

VLOG returns an exit code of 0 if successful, 1 otherwise.

Syntax

```
vlog [option...] file_name...
```

Use the *file_name* parameter to specify one or more archives or workfiles. File specifications can include wildcards.

The following table lists frequently used commands. See the "Options," section for details on all command-line options.

To view...	Use this command...
The archive name, user ID of the locker, locked revision number, and check-in revision number	VLOG -BL
A one-line report on newest revisions	VLOG -BN
Information about the specified revisions	VLOG -BR

Default versions

The DefaultVersion directive affects the following commands:

VLOG -BN
VLOG -BR
VLOG -BV
VLOG -R
VLOG -V

You can override this directive by using a hyphen (-) after these options. When you override the DefaultVersion directive, you see information for the tip revision of the trunk.

Advanced label filtering

If AdvancedLabelFilter directive (set by default since 8.7.0) is set then advanced label filter processing take place.

Allowed operation:

! - NOT

& -AND

| - OR

()

''

""

Wildcards:

* -any string or empty string

? -any char

[] - set of allowed chars

[^] - set of not allowed chars

Start filter with:

'any:' - match if there any label in revision matched filter

'all:' - match if there are all labels in revision matched filter

Examples:

vlog -r"all:('lab* ' | *my?) & !*ver[^123]*" D:\pdbs\pdb2\archives\vr11d.c-arc

Options

@ `@[list_file]`

Use the `@` option to read *`list_file`* for additional command-line options. Version Manager reads additional options from the file before reading the rest of the command line. Use the `@` option with no parameter if you're redirecting input from another command.

If the file has the extension .GRP, you can omit the extension and the path. Version Manager searches for it in the directories specified by the VCSDir directive.

-# `-#code`

Use the -# option to display diagnostic information for debugging purposes. Use the *`code`* parameter to select one of the following diagnostic functions:

`1`	Display each line of the configuration file as it is processed.
`2`	Display internal errors.
`4`	Toggle the sign-on message.
`500`	Display the archive owner, access list, access control database name, current user ID, and their privileges. The currently active privileges are in uppercase letters.

To specify more than one diagnostic item, add the numbers together. For example, use -#`3` to display the diagnostic information represented by numbers `1` and `2`.

The -# option should be the first option after the command. Version Manager processes commands from left to right and does not display debugging information until it reaches the -# option.

-a `-auser_id[,user_id...]`

Produces a report containing only changes made by certain users. Separate multiple user IDs with commas.

-b Displays archive information but not revision information.

-bc `-bcrevision_number | version_label`

Produces a report of archives whose tip revisions are not the same as the specified revision number or version label. This feature is useful when you want to list all archives that have changed since a particular version. Note, if the archive does not include the specified label, it is not included in the report.

-bg `-bgpromotion_group`

Produces a brief report showing which revisions are associated with the specified promotion group.

-bl `-bl[user_id[,user_id...]]`

Produces a report of currently locked revisions, showing the archive name, user ID of the locker, revisions locked, and check-in revision numbers. Use the optional *user_id* parameter to limit the report to revisions locked by that user. Separate multiple user IDs with commas.

-bn `-bn[`*branch*`]`

Produces a one-line report of the newest revision on the specified *branch*. If you don't specify a *branch*, you see the report for the trunk.

-br `-br[`*revision_number* | *revision_range* | *branch*`]`

Produces a report of revision information. The report does not include information about the archive itself, such as header or attribute information. If you supply the *revision_number* or *revision_range* parameter, VLOG displays information about the specified revision(s) only.
`-br`*version_label* | *version_range*

Use this form of the `-br` option to see revision information for revisions associated with the specified version label, or with revisions within and including a range of version labels.

-bv `-bv[`*version_label*`]`

Produces a report of the revision numbers that correspond to the specified version label. This option is a convenient way to obtain a list of revision numbers that are associated with a given version label.

If you don't specify a version label, the report shows information about all version labels in each specified archive, unless DefaultVersion is defined.

-c `-c[- |`*configuration_file*|*directory_name*`]`

Instructs Version Manager to look for configuration information in a place other than in the master configuration file or the default configuration file.

See "Configuration Files" on page 111 for details on using the `-c` option.

-d `-d`*date_range*

Limits the report to revisions checked in within the specified range of dates.

-g `-g`*promotion_group*

Restricts the report to revisions associated with the specified promotion group.

-h Displays help for command-line options. The program terminates after it processes this option even if you specify other options.

-i Turns off the indentation used in the report to identify branch revision information.

-id **`-id`***user_id*`:`*user_password*

(File server only.) Specifies a user ID and password to authenticate against the access control database and/or LDAP server, if any, that is associated with the relevant file server path map in the Version Manager File Server Administration utility.

NOTE The `-id` option does not affect the user ID used to perform the CLI operation itself. The user ID for the CLI operation is determined in the normal manner.

-l `-l[`*user_id*`[,`*user_id*`...]]`

Lists the archives that contain locks. To list locked revisions, use VLOG -BL, or use the optional *user_id* parameter to limit the report to revisions locked by specific users. Separate multiple user IDs with commas.

-o `-ouser_id[user_id...]`

Limits the report to archives owned by certain users. Separate multiple user IDs with commas.

-q Selects the quiet mode of operation, in which the program displays the report and any error messages.

-r `-r[revision_number | revision_range]`

Specifies the revision number or range of revisions to be reported. The range is specified using any combination of revision numbers. If you use -r without a revision number, the program uses the latest trunk revision, unless DefaultVersion is defined. If the version label begins with a number, precede it with a backslash (\).

-s `-ssuffix_template`

Specifies the archive suffix template to use instead of the current value of ArchiveSuffix.

-u Displays the current user's ID and the source of the user ID (such as Netware, etc.).

-v `-vversion_label | version_range [+]`

Specifies revisions to be reported on using version labels. To get a report on all revisions between two version labels, enter a range of version labels.

To report on all branches originating from a revision or range of revisions, follow the version label or range with a plus sign (+).

-xe `-xefile_name`

Redirects status, program, and error messages to *file_name*.

-xo `-xofile_name`

Redirects standard output to *file_name*.

-xo+e `-xo+efile_name`

Redirects standard output and error messages to *file_name*.

Examples

- The following example produces a complete report for the archive TEST.C_V:
  ```
  vlog test.c_v
  ```
- The following example produces a brief report for the archive associated with workfile TEST.C:
  ```
  vlog -b test.c
  ```
- The following example produces a brief report of the revisions contained in the archive TEST.C_V:
  ```
  vlog -br test.c_v
  ```
- The following example produces a list of all locked revisions in all archives:
  ```
  vlog -bl *.??v
  ```
- The following example produces a complete report for all archives owned by user bonnier whose workfiles have the extension .C:
  ```
  vlog -obonnier *.c_v
  ```

Special Considerations

- If you use multiple options that restrict the report, the resulting report shows the intersection of the options. For example, the following command produces a report of revisions checked in during January to archives owned by user colette:
  `vlog -dJanuary/1992 -ocolette *.??v`
- The VJOURNAL -XL command produces the same report of locked revisions as the VLOG -L command. VJOURNAL is much faster, because it does not have to read each archive to obtain the information. However, the VJOURNAL report may be inaccurate or unavailable. Causes of inaccuracy include users with inconsistent configuration files, locked archives being moved, and the journal file being unavailable while revisions are being locked or unlocked.

Related Topics

For information about...	See...
Defining a default revision or version	"DefaultVersion directive" on page 135
Related commands	"VJOURNAL command" on page 73
Other Version Manager commands	"Using Commands" on page 8

VMRG command

Merge two sets of revisions to a common base

Use the VMRG command to compare two files to a common ancestor and combine the changes they represent into a single new revision.

Privileges required. This command requires the Get privilege to check out revisions from archives.

Exit code

VMRG returns an exit code of 0 if successful, 1 otherwise.

Syntax

`vmrg [option...] parent_revision branch_point1 branch_point2`

Specify the *parent_revision*, *branch_point1*, and *branch_point2* parameters in one of the following ways:

Use this form...	To specify this...
file_name	A text file or workfile
archive	The latest trunk revision in an archive
`-rrevision` *workfile* | *archive*	That revision of the archive
`-vversion` *workfile* | *archive*	That version of the archive

The following table lists frequently used commands. See the "Options," section for details on all command-line options.

To do this...	Use this option...
Specify more than one file to be merged	vmrg -m
Specify the name of the output file	vmrg -o

Default versions

The DefaultVersion directive affects the following commands:

vmrg -r
vmrg -v

You can override this directive by using a hyphen (-) after the option. When you use a hyphen to override the DefaultVersion directive, you operate on the tip revision of the trunk.

Specifying Files to Merge

Determine unspecified file names

If both *branch_point1* and *branch_point2* are revision numbers, VMRG uses the first archive name it encounters on the command-line for both revisions. For example, the following commands are equivalent:

```
vmrg -r1.3 -r1.3.1.7 -r error.c_v
    vmrg -r1.3 error.c_v -r1.3.1.7 -r error.c_v
```

If you don't specify one of the *workfile* or *archive* parameters, the program assumes that it is the same as the one implied by the parameter that you did specify.

Specify the merge file output

If *branch_point2* is a workfile name, Version Manager uses it as the output file, or merge file, and prompts you for permission before overwriting it. You can avoid overwriting *branch_point2* by using the -o option. For example, to capture the merged output as a file called MERGE.OUT, you would type:

vmrg -omerge.out -r1.3 -r1.3.1.7 -r error.c

Default merge file

If you do not specify an output file and *branch_point2* is an archive, VMRG uses the workfile name in the archive as the output file.

Contents of the merge file

If an area of the *parent_revision* was changed in both *branch_point1* and *branch_point2*, VMRG sends both sets of changes to the merge file. These conflicts are set off from the rest of the text as shown below:

```
 >>>>>>>>>> branch_point1
:
:(lines from branch_point1)
:
<<<<<<<<<< branch_point2
:
:(lines from branch_point2)
:
==========
```

VMRG displays a message, warning you to edit the resulting file to choose the correct set of changes.

Options

@ `@[list_file]`

Use the `@` option to read *list_file* for additional command-line options. Version Manager reads additional options from the file before reading the rest of the command line. Use the `@` option with no parameter if you're redirecting input from another command.

If the file has the extension .GRP, you can omit the extension and the path. Version Manager searches for it in the directories specified by the VCSDir directive.

-# `-#code`

Use the -# option to display diagnostic information for debugging purposes. Use the *code* parameter to select one of the following diagnostic functions:

1	Display each line of the configuration file as it is processed.
2	Display internal errors.
4	Toggle the sign-on message.
500	Display the archive owner, access list, access control database name, current user ID, and their privileges. The currently active privileges are in uppercase letters.

To specify more than one diagnostic item, add the numbers together. For example, use -#3 to display the diagnostic information represented by numbers 1 and 2.

The -# option should be the first option after the command. Version Manager processes commands from left to right and does not display debugging information until it reaches the -# option.

-c `-c[- | configuration_file | directory_name]`

Instructs Version Manager to look for configuration information in a place other than in the master configuration file or the default configuration file.

See "Configuration Files" on page 111 for details on using the -c option.

-g `-gpromotion_group`

Specifies the revision associated with a particular promotion group.

-h Displays help for command-line options. The program terminates after it processes this option even if you specify other options.

-id **-id***user_id*:*user_password*

(File server only.) Specifies a user ID and password to authenticate against the access control database and/or LDAP server, if any, that is associated with the relevant file server path map in the Version Manager File Server Administration utility.

NOTE The -id option does not affect the user ID used to perform the CLI operation itself. The user ID for the CLI operation is determined in the normal manner.

-m Merges multiple files by specifying more than one workfile or archive. By specifying a version label as the *parent_revision* parameter, and version labels for the *branch_point1* and *branch_point2* revision, you can merge many files with the same command. Furthermore, by using wildcards, you can specify multiple archives in a merge of two or more branches.

-n Answers no in advance to any question VMRG may ask you.

-o `-o[file_name]`

Redirects the merged file to *file_name*. Omit the *file_name* parameter to send the merged file to standard output.

-r `-r[revision_number]`

Specifies the revision number to be merged. If you use -r without a revision number, the program uses the latest trunk revision, unless DefaultVersion is defined. If the version label begins with a number, precede it with a backslash (\).

-s `-ssuffix_template`

Specifies the archive suffix template to use instead of the current value of ArchiveSuffix.

-v `-v[version_label]`

Specifies the version label assigned to *branch_point1* or *branch_point2*. If you defined the DefaultVersion directive, you can omit the *version_label* parameter.

-xe `-xefile_name`

Redirects status, program, and error messages to *file_name*.

-xo `-xo[file_name]`

Redirects all program messages to standard output to the file *file_name*.

-xo+e `-xo+efile_name`

Redirects standard output and error messages to *file_name*.

-y Answers yes in advance to any question VMRG may ask you.

Special Consideration: You can only use the `-m` option if all revisions are contained in archives.

Related Topics

For information about...	See...
Defining a default revision or version	"DefaultVersion directive" on page 135
Other Version Manager commands	"Using Commands" on page 8

VPROMOTE command

Move revisions up in the promotion hierarchy: Use the VPROMOTE command to move a revision from one promotion group to the next higher promotion group.

To define the hierarchy of promotion groups, you use the Promote directive.

Syntax: `vpromote [option...]-gpromote_from_group file_name...`

Options

@ `@[list_file]`

Use the `@` option to read *list_file* for additional command-line options. Version Manager reads additional options from the file before reading the rest of the command line. Use the `@` option with no parameter if you're redirecting input from another command.

If the file has the extension .GRP, you can omit the extension and the path. Version Manager searches for it in the directories specified by the VCSDir directive.

-# `-#code`

Use the -# option to display diagnostic information for debugging purposes. Use the *code* parameter to select one of the following diagnostic functions:

1	Display each line of the configuration file as it is processed.
2	Display internal errors.
4	Toggle the sign-on message.
500	Display the archive owner, access list, access control database name, current user ID, and their privileges. The currently active privileges are in uppercase letters.

To specify more than one diagnostic item, add the numbers together. For example, use -#3 to display the diagnostic information represented by numbers 1 and 2.

The -# option should be the first option after the command. Version Manager processes commands from left to right and does not display debugging information until it reaches the -# option.

-c `-c[- | configuration_file | directory_name]`

Instructs Version Manager to look for configuration information in a place other than in the master configuration file or the default configuration file.

See "Configuration Files" on page 111 for details on using the -c option.

-g `-gpromote_from_group`

Specifies the promotion group from which you want to promote.

Privileges required. The -g option requires the Promote privilege when specifying promotion groups.

-h Displays help for command-line options. The program terminates after it processes this option even if you specify other options.

-id **-id***user_id*:*user_password*

(File server only.) Specifies a user ID and password to authenticate against the access control database and/or LDAP server, if any, that is associated with the relevant file server path map in the Version Manager File Server Administration utility.

NOTE The -id option does not affect the user ID used to perform the CLI operation itself. The user ID for the CLI operation is determined in the normal manner.

-q Selects the quiet mode of operation, in which Version Manager only reports error conditions. This option suppresses messages and queries.

-qo Directs Version Manager to report only error codes, but asks permission before overwriting existing files. When you use -qo with the -y option, it is equivalent to -q.

-xe `-xefile_name`

Redirects status, program, and error messages to *file_name*.

-xo `-xofile_name`

Redirects standard output to *file_name*.

Related Topics

For information about...	See...
Creating a promotion model	"Quiet directive" on page 164

VSPLIT command

Separate revisions and metadata

Use the VSPLIT command to split existing inflated archives into separate revision and metadata stores for use with the revision library feature of the Version Manager File Server. VSPLIT can also unsplit archives in revision libraries to return them to the original inflated archive format.

Use this feature in conjunction with a Version Manager File Server.

Syntax

vsplit [*Options*] *Path* . . .

Where *Path* is the Client Name that represents the project database as defined in the Path Maps pane of the Version Manager File Server Administration Utility.

Options

@ `@[list_file]`

Use the `@` option to read *list_file* for additional command-line options. Version Manager reads additional options from the file before reading the rest of the command line. Use the `@` option with no parameter if you're redirecting input from another command.

If the file has the extension .GRP, you can omit the extension and the path. Version Manager searches for it in the directories specified by the VCSDir directive.

-h Displays help for command-line options. The program terminates after it processes this option even if you specify other options.

-id **-id***user_id*:*user_password*

(File server only.) Specifies a user ID and password to authenticate against the access control database and/or LDAP server, if any, that is associated with the relevant file server path map in the Version Manager File Server Administration utility.

NOTE The `-id` option does not affect the user ID used to perform the CLI operation itself. The user ID for the CLI operation is determined in the normal manner.

-l Lists the archives but does not split them.

-r Recursive (applies to directory paths).

-u Unsplit recombines revision and metadata from split archives to return them to the original inflated archive format. The reconstituted archive is located in the metadata location. The Revision Library is deleted.

TIP Disable the RFSSplitOnCreate directive or any new files you add will end up in split archives.

-v Verbose mode.

-xe `-xe`*`file_name`*

Redirects status, program, and error messages to *`file_name`*.

-xo `-xo`*`file_name`*

Redirects standard output to *`file_name`*.

Special Considerations

- VSPLIT resides on the client and references the archive path on the Version Manager File Server as viewed by the client.
- VSPLIT affects only existing archives. It does not split new archives. Use the RFSSplitOnCreate directive to split archives when they are created.
- This command has no effect unless the archives are mapped to a Version Manager File Server.
- You can run the VSPLIT command only if it is enabled by the **Enable Utilities** option in the Version Manager File Server Administration Utility.
- The VSPLIT command is located in the bin\admin directory of your Version Manager installation. See your Version Manager administrator if you need to use this command but do not have access to it.

VSPLIT: Usage Suggestions

We recommend that you do the following:

- **Disable User Access:** If the archives to be split are already available to users, disable or limit user access to the archives before running VSPLIT. Once the operation is complete and you have verified the results, you can re-enable user access. If you will run VSPLIT from the file server itself and the archives are local to the server, you can disable user access simply by disconnecting the server from the network.
- **Test:** Set up a test environment that matches your actual production environment as closely as possible. Use this test environment to test the upgrading process against a copy of your project databases.
- **Back Up:** Back up your data before running VSPLIT.
- **Redirect Output:** Redirect the output of the VSPLIT command to a file so you can analyze the output to identify any archives that were not properly split. To redirect the output to a file, use the **`-xo`** and **`-xe`** options, which can be combined as **`-xo+e`**. For example:
  `vsplit -xo+e``OutPut.txt` **`-r`** `S:\VMFS\PDBS`

Analyzing VSPLIT Output

VSPLIT checks each archive before and after it is split. It will not split corrupted archives or any non-archive files it encounters.

Various non-archive files (access control databases, journal files, and configuration files) will be listed as conversion failures—this is expected and is of no concern. You should examine the output for any archives that failed.

An entry is made for each file. The following example shows an archive that was successfully split and a journal file which failed—as expected—since it is not an archive and cannot be split.

```
convert(S:\VMFS\PDBS\archives\test\env_filt.sed-arc) ok 733/1425 bytes
    (new/old)
vcheck(S:\VMFS\PDBS\archives\test\journal.vcs)=1 failed, not converting
```

If unexpected failures appear in the VSPLIT output, examine the `pvcsfs.log` file. Errors recorded in this file may help you resolve the underlying problem. For information about viewing the file server log, see the *Version Manager Administrator's Guide*.

Related Topics

For information about...	See...
Version Manager File Servers	*Administrator's Guide*
RFSSplitOnCreate directive	"RFSSplitOnCreate directive" on page 168
VTRANSFER command	"VTRANSFER command" on page 98

VSQL command

Call the SQL Exporter

Use the VSQL command to transfer archive information into SQL (Structured Query Language) relational database systems. You can then use standard SQL queries to produce project documentation and reports.

To use this feature, you must have a third-party SQL database system, such as Gupta, Oracle, or XDB, and a basic knowledge of SQL and relational database concepts.

Privileges required. This command requires the ViewArchive privilege.

Syntax

```
vsql [option...] file_name...
```

Use the *file_name* parameter to specify the archives that contain data to be transferred to the database.

If you use a single or double quotation mark (' or ") as part of any VSQL option, precede it with a backslash (\). If a character string contains spaces, enclose it with single or double quotation marks.

The following table lists frequently used commands. See the "Options," section for details on all command-line options.

To do this...	Use this command...
Define a table format	vsql -d
Generate database files in different formats	vsql -m
Generate SQL DROP statements	vsql -r
Generate update statements	vsql -u

Recommended Options

Use options for different databases

Use the following options for smooth translation into the formats used by different database systems. You may want to place the options for your database system in a command file and use the Include directive with this file.

Database System	File Format	Options
XDB	SQL*	-K250
	CSV	-O\"
Gupta	SQL	-DV"LONG VARCHAR(1500)"
	CSV	-O\" -DV"LONG VARCHAR(1500)"
SQL Server	SQL	-DI"INT" -DV"TEXT(1500)" -DT"CHAR(20)"
	CSV	-O\" -DI"INT" -DV"TEXT(1500)" -DT"CHAR(20)"
Oracle	SQL	-DV"LONG VARCHAR(1500)" -DT"CHAR(20)"
	CSV	-O\" -DV"LONG VARCHAR(1500)" -DT"CHAR(20)"

*.For DDL and DML

SQL DDL Command Files

Generate DDL commands

You can use VSQL to generate Data Definition Language (DDL) commands to create or redefine relational tables and views for storing the archive data. The files can optionally contain DROP commands to drop any previously defined Version Manager tables and views.

The following list shows the tables and views that appear on subsequent pages:

SolvCTAccs Information about archive access privileges

SolvCTBrns Information about branches

SolvCTGrps Information about promotion groups

SolvCTLogs Information pertaining to the entire archive

SolvCTLoks Information about locked revisions

SolvCTRevs Information about all archive revisions

SolvCTVers Information about version labels

Column Name	Default Type	Default Size (Bytes)	Description
ArchiveID	INTEGER	4	Surrogate archive ID
Name	CHAR	64	User ID

Column Name	Default Type	Default Size (Bytes)	Description
ArchiveID	INTEGER	4	Surrogate archive ID
Revision	CHAR	48	Revision number
Branch	CHAR	48	Branch number

Column Name	Default Type	Default Size (Bytes)	Description
ArchiveID	INTEGER	4	Surrogate archive ID
Revision	CHAR	48	Revision number
GroupName	CHAR	64	Promotion group associated with this revision

Column Name	Default Type	Default Size (Bytes)	Description
ArchiveID	INTEGER	4	Surrogate archive ID
Archive	CHAR	254	Archive name and path
Workfile	CHAR	254	Workfile name
ArchiveOwner	CHAR	64	Archive owner ID
RevisionCount	SMALLINT	2	Number of revisions in archive
LastModified	TIMESTAMP	*	Time and date of last archive modification
Secure	CHAR	1	Archive secure (Y/N)
WriteProtect	CHAR	1	Archive write-protected (Y/N)
CheckLock	CHAR	1	Lock checking enabled (Y/N)
ExclusiveLock	CHAR	1	Exclusive lock enforced (Y/N)
ExpandKeyword	CHAR	1	Keyword expansion enabled (Y/N)
TranslateEOL	CHAR	1	EOL translator enabled (Y/N)
CompressDelta	CHAR	1	Compress deltas (Y/N)
CompressWork	CHAR	1	Compress text (Y/N)
GenerateDelta	CHAR	1	Delta generation enabled (Y/N)

Column Name	Default Type	Default Size (Bytes)	Description
CommentPrefix	CHAR	16	Line prefix string for expanded keywords
Newline	CHAR	16	Line termination string for expanded keywords
Description	VARCHAR	1500	Archive description

* System-dependent

Column Name	Default Type	Default Size (Bytes)	Description
ArchiveID	INTEGER	4	Surrogate archive ID
Revision	CHAR	48	Number of the locked revision
RevToPut	CHAR	48	Revision that will be created at check-in
Locker	CHAR	64	Locker ID

Column Name	Default Type	Default Size (Bytes)	Description
ArchiveID	INTEGER	4	Surrogate archive ID
Revision	CHAR	48	Revision number
Author	CHAR	64	Author ID
ModifyDate	TIMESTAMP	*	Date and time of modification
CheckInDate	TIMESTAMP	*	Date and time of check-in
Deleted	INTEGER	4	Number of lines deleted
Inserted	INTEGER	4	Number of lines inserted
Moved	INTEGER	4	Number of lines moved
RevisionNote	VARCHAR	1500	Revision update note

*System-dependent

Column Name	Default Type	Default Size (Bytes)	Description
ArchiveID	INTEGER	4	Surrogate archive ID
Revision	CHAR	48	Revision number
Version	CHAR	128	Version label
Floating	CHAR	1	Floating version label (Y/N)

About the Tables and Views

Archive ID. The first column in all tables and views is ArchiveID. The ArchiveID column provides a unique identification based on the time the archive was read and the database information generated. It is unique in the table SolvCTLogs and provides links to all other tables. It is meant to be used in "join" clauses of SQL queries.

Indexing. Since the syntax of key and indexing specification varies widely among commercial SQL systems, the DDL statements do not attempt to generate any indexes and do not specify key columns.

We recommend that you index at least the database tables on ArchiveID, or specify the ArchiveID in SolvCTLogs as a primary (or unique) key and ArchiveID in all other tables as a foreign (or non-unique) key.

Views. The views correspond to the tables as follows:

View	Corresponding Table
SolvCVAccs	SolvCTAccs
SolvCVBrns	SolvCTBrns
SolvCVGrps	SolvCTGrps
SolvCVLogs	SolvCTLogs
SolvCVLoks	SolvCTLoks
SolvCVRevs	SolvCTRevs
SolvCVVers	SolvCTVers

SQL Formats

Generate files in different formats

You can use the VSQL command to generate files containing data about specified archives in the following formats:

- **SQL DML Format.** Use SQL DML (Data Manipulation Language) to generate an output file containing Insert statements. Your SQL interpreter can interpret this file in the same manner as the DDL command file, described above.
- **CSV Format.** Use the CSV (Comma Separated Variable) option to generate one output file for each table (six files total). The files have variable ASCII formats with fields separated by field separator characters. You can enclose non-numeric fields in quotation marks (").

You can also generate an optional header record that contains column names.

The program infers the name of the generated file from the archive name by adding or replacing the file extension as follows:

Table	File name
SolvCTAccs	file.ACC
SolvCTBrns	file.BRN
SolvCTGrps	file.GRP
SolvCTLogs	file.LOG

Table	File name
SolvCTLoks	file.LOK
SolvCTRevs	file.REV
SolvCTVers	file.VER

Generating Command Files

Produce files used by the database system

The following example assumes you are using the XDB (DB2-compatible) database system. You can easily modify this procedure for other products.

1. To generate the DDL command file, type:
   ```
   vsql -K250 -DL"ddl.sql"
   ```
2. To generate the DML command file, type:
   ```
   vsql -K250 -ML"dml.sql" \archives\*.*
   ```
   Alternatively, to generate a CSV format file with a header, type:
   ```
   vsql -O\" -MH"csv" \archive\*.*
   ```
3. First use your database interface system to execute DDL.SQL. Then run either DML.SQL or the database import or load utility for all CSV.* files that were generated.
4. Write and execute SQL queries to test the reporting capabilities.

Options

@

`@[list_file]`

Use the `@` option to read *`list_file`* for additional command-line options. Version Manager reads additional options from the file before reading the rest of the command line. Use the `@` option with no parameter if you're redirecting input from another command.

If the file has the extension .GRP, you can omit the extension and the path. Version Manager searches for it in the directories specified by the VCSDir directive.

-#

`-#code`

Use the -# option to display diagnostic information for debugging purposes. Use the *`code`* parameter to select one of the following diagnostic functions:

`1`	Display each line of the configuration file as it is processed.
`2`	Display internal errors.
`4`	Toggle the sign-on message.
`500`	Display the archive owner, access list, access control database name, current user ID, and their privileges. The currently active privileges are in uppercase letters.

To specify more than one diagnostic item, add the numbers together. For example, use -#`3` to display the diagnostic information represented by numbers `1` and `2`.

The -# option should be the first option after the command. Version Manager processes commands from left to right and does not display debugging information until it reaches the -# option.

-c `-c[- | configuration_file | directory_name]`

Instructs Version Manager to look for configuration information in a place other than in the master configuration file or the default configuration file.

See "Configuration Files" on page 111 for details on using the `-c` option.

-di `-diinteger`

Specifies an alternative to the INTEGER column-type definition, depending on the requirements of your SQL interpreter.

-dl `-dl[file_name]`

Generates a file that uses SQL DROP TABLE (optionally, see below), CREATE TABLE, and CREATE VIEW commands to initiate tables for holding Version Manager archive data and views used for updating these tables. If you don't specify the *file_name* parameter, the program displays the file.

This option must be executed by the SQL interpreter first to provide tables for Version Manager data transfer.

When the `-dl` option is used together with `-r` option, SQL DROP TABLE and DROP VIEW commands are generated before the CREATE commands. This option is useful to re-execute the command file. It destroys and recreates the old tables and views.

Your SQL system may require the SQL command files to have a particular extension.

-dt `-dttimestamp`

Specifies an alternative to the TimeStamp column type definition, depending on the requirements of your SQL interpreter.

-dv `-dvvarchar_def`

Defines an alternative for the long variable character column type definition. VARCHAR(1500) is the default.

-e `-e[character]`

Specifies the character that the program substitutes for the string delimiter character when it is embedded within a string. The default string delimiter is single quotation marks.

To use a single quotation mark (') or double quotation mark ("), preface it with a backslash (\). To use a backslash, enter two backslashes.

For example, if your string delimiter is a single quotation mark ('), and you expect to use single quotation marks within strings, you can use -E\" (double quotation mark) to ensure that the program won't confuse string delimiter characters with characters within strings.

-ft `-fttime_format`

Specifies an alternative format for the timestamp fields, or the fields designated by the `-dt` option. The *time_format* parameter can contain the special formatting strings below.

String	Meaning
yyyy	A four-digit number representing the year portion of the date. You can also use *yy* to represent the last two digits of the year.
mm	(First occurrence.) A two-digit number between 01 and 12 representing the month. If you enter more than two consecutive *m* characters, the program interprets them as an abbreviation of the month. For example, *mmm* = Jun; *mmmm* = June.
dd	A two-digit number between 01 and 31 (the maximum depends on the month and year) representing the day of the month.
hh	A two-digit number between 00 and 24 representing the hour portion of the time (using a 24-hour clock).
mm	(Second occurrence.) A two-digit number between 00 and 59 representing the minutes portion of the time.
ss	A two-digit number from 00 to 59 representing the seconds portion of the time.

If you specify formatting strings that contain other than the specified number of consecutive formatting characters, the program copies them to the resulting timestamp without change. The treatment of months is an exception to this rule, as explained above.

If the format string starts with "-noquotes:" then the output string will be without quotes.

The default timestamp formatting string is yyyy-mm-dd-hh.mm.ss.

-g `-g[`*`character`*`]`

Specifies a field separator character. The default field separator is a comma (,).

-h Displays help for command-line options. The program terminates after it processes this option even if you specify other options.

-i `-i[`*`character`*`]`

Specifies the character that the program substitutes for the field separator character when it is embedded within a field.

For example, if your field separator is a comma, and you expect to use commas within strings, use -I; (or another non-conflicting character) to change the treatment of commas within a field.

The `-i` option is usually needed only in CSV format files if no string delimiter is used.

-id **`-id`***`user_id`*`:`*`user_password`*

(File server only.) Specifies a user ID and password to authenticate against the access control database and/or LDAP server, if any, that is associated with the relevant file server path map in the Version Manager File Server Administration utility.

NOTE The `-id` option does not affect the user ID used to perform the CLI operation itself. The user ID for the CLI operation is determined in the normal manner.

-j `-j[`*`character`*`]`

Specifies the line continuation character. The default is an empty string. For example, if you want to use a backslash (\) to specify line continuation, use `-J\`.

If you use the –j option to specify a line continuation character, you may also need to use the –l option to specify a replacement for the line continuation characters found inside string fields.

-k –k*number*

Specifies the maximum line length for the output command files. The default is an unlimited length. For example, use -k250 to limit the line length to 250 characters, including delimiters and modification characters.

The minimum line length is 64 characters. If you use a number smaller than 64, the line length will default to 64 characters.

-l –l[*character*]

Specifies the character that the program substitutes for the line continuation character when it is embedded within a string.

The default is an empty string.

-mh –mh*file_name*

Generates a set of CSV-format files containing data from the specified archives. The column names are the first record of each file. This information is used by import utilities in some database systems. The program infers the output file name from the archive name by adding or replacing the extension with .LOG, .LOK, .REV,.GRP., .VER, .ACC, and .BRN. You can then import the files to the database using an import utility usually provided with the database system. The header line, including a title for each field precedes the data line. This is different from -MV where the header lines are not generated.

-ml –ml[*file_name*]

Use the –ml option to generate SQL DML format Insert statements. If you don't enter the *file_name* parameter, the program displays the resulting file.

-mv –mv*file_name*

Use the -mv option to generate a set of CSV format files containing data from the specified archives. This option differs from -mh in that it does not include column headings. The program infers file names from the specified *file_name* by adding P., .VER, .ACC, and .BRN. This is different from –mh; with –mv VSQL generates data lines without header lines.

You can then import the files to the database using an import utility usually provided with the database system. The program creates all script files; however, they may be empty.

-n –n[*character*]

Specifies the character that the program substitutes for the newline character when it is embedded within a string. By default, the program substitutes a space. If you don't use this option, strings cannot contain newline characters.

-o –o[*character*]

Specifies the string delimiter. Most CSV formats are processed with the least conflict using double quotation marks as the string delimiter. The default, however, is a single quotation mark to accommodate the DML format.

To use a single quotation mark (') or double quotation mark ("), preface it with a backslash (\). To use a backslash, enter two backslashes.

-q Selects the quiet mode of operation, in which Version Manager only reports error conditions. If you don't redirect DLL or DML output to a file, the program still displays that output when you use this option.

-r Together with the `-dl` option, generates SQL DROP statements in the DDL command file. DROP commands remove tables and views from the database prior to generating the new tables and views. This option is useful if you need to rerun the DDL command file with the old tables still defined in the database.

-s `-s`*`suffix_template`*

Specifies the archive suffix template to use instead of the current value of ArchiveSuffix.

-t `-t[`*`character`*`]`

Specifies the SQL statement termination character. The default is a semicolon (;). If your SQL interpreter requires no statement terminator, use the `-t` option with no *`character`* parameter.

-u Generates update statements. When you use the -u option together with the `-ml`, `-mv`, or `-mh` option, the program generates DML Delete statements to delete the old information about the archive before inserting the new data into the tables.

If you use the `-u` option with the `-ml` option, the program places the Delete statements in the same file as the rest of the DML statements. If you use the `-u` option with the `-mv` or `-mh` option, it places the Delete statements in a separate file with the name specified by the `-mv` or `-mh` option. (Remember that the CSV format output uses only the base part of the specified name and appends new extensions corresponding to the relational table in question.)

This option can be useful to maintain an existing database of archives on an incremental basis; however, in some cases it may be just as easy to delete the tables and regenerate the whole database from scratch.

-xe `-xe`*`file_name`*

Redirects status, program, and error messages to *`file_name`*.

-xo `-xo`*`file_name`*

Redirects standard output to *`file_name`*.

-xo+e `-xo+e`*`file_name`*

Redirects standard output and error messages to *`file_name`*.

Sample SQL Queries

Sample database queries

- To see general information about archives, type:

```
SELECT * FROM SolvCVLOGs;
```

- To see workfile names and revision numbers of all locked workfiles and the corresponding locker IDs, type:

```
SELECT Workfile, Revision, Locker
FROM SolvCVLOGs, SolvCVLoks
WHERE SolvCVLOGs.ArchiveID = SolvCVLoks.ArchiveID;
```

- To see workfile names of all files modified by user John, type:

```
SELECT DISTINCT Workfile
FROM  SolvCVLOGs, SolvCVRevs
WHERE SolvCVLOGs.ArchiveID = SolvCVRevs.ArchiveID
AND Author = 'John';
```

Special Considerations

- Your operating system command interpreter may treat characters such as >, &, and | as special symbols. You cannot use these characters in VSQL command-line options.
- All SQL modification statements insert to views rather than tables, which allows you more flexibility in adding custom columns and making future changes to the structure of the database.
- You must execute the DDL command file using the SQL interpreter before you attempt to import any data to the database.
- The program truncates all strings retrieved from archives to the maximum lengths of their columns.
- Your SQL system may require SQL command files to have a particular extension.
- Archive IDs are calculated from the current time. To avoid duplicate IDs, do not run multiple instances of VSQL in quick succession.

Related Topics

For information about...	See...
Other Version Manager commands	"Using Commands" on page 8

VTRANSFER command

Export, import, move, rename, fixarchives

VTRANSFER is a command-line utility for use with the Version Manager File Server that allows you to:

- Export the metadata and revision data of an archive, including split archives, as a single *.zip file. This is a convenient way to gather and package an archive so it can be sent to Micro Focus support for troubleshooting.
- Import the metadata and revision data of an archive, including split archives, from a single *.zip file. This is a convenient way to import changes made to troubleshoot an archive.
- Move an archive from one location to another on the same server.
- Rename an archive.
- Fix the relationship between the metadata and revision data (revision library) if the archive has been manually moved or renamed.
- Delete an archive, both the metadata and revision data.

Syntax

vtransfer [*options*]*archivePath* [*secondaryPath*]

Where:

- *archivePath* is the location of the archive you are operating on.
- *secondaryPath* is a parameter specific to the mode in which vtransfer is being used, such as the location of the *.zip file or a new name for the archive.

Options

VTRANSFER has six *mode options*, which determine what sort of operation it is to perform, as well as ten general options.

Mode Options

-c Copies an archive or a directory of archives to the name/location specified in the *secondaryPath*. This operation is recursive if the *archivePath* specifies a directory rather than an archive.

-d Deletes the archive or a directory of archives specified in the *archivePath*, including all revisions and metadata.

-f Fixes the relationship between the metadata and revision data by moving the revisions to a location that mirrors the metadata location, and then updates the metadata. This is useful if the metadata has been manually moved or renamed.

-i Imports an archive and its revisions from a zip file specified in the *secondaryPath* to the archive location specified in the *archivePath*. Any existing archive data is overwritten.

-r Renames or moves an archive or a directory of archives to the name/location specified in the *secondaryPath*. This operation is recursive if the *archivePath* specifies a directory rather than an archive.

-x Exports the metadata and revision data of the archive specified in the *archivePath* and writes it to a single *.zip file as specified in the *secondaryPath*. If the *secondaryPath* does not contain a fully qualified path, the file is written to the location from which you ran the command. The revisions are exported from the path implied by the revision path mapped on the file server and the *archivePath*.

Use with the **-m** option to export revisions from the location specified in the metadata. This is useful if the metadata has been manually moved.

General Options

@ `@[list_file]`

Use the `@` option to read *list_file* for additional command-line options. Version Manager reads additional options from the file before reading the rest of the command line. Use the `@` option with no parameter to redirect input from another command.

If the file has the extension .GRP, you can omit the extension and the path. Version Manager searches for it in the directories specified by the VCSDir directive.

-h Displays help for the command. The command terminates after processing the **-h** option even if you specify other options.

-id **-id***user_id*:*user_password*

Specifies a user ID and password to authenticate against the access control database and/or LDAP server, if any, that is associated with the relevant file server path map in the Version Manager File Server Administration utility.

NOTE The `-id` option does not affect the user ID used to perform the CLI operation itself. The user ID for the CLI operation is determined in the normal manner.

-m Use with **-x** or **-f** to export revisions from the location specified in the metadata. This is useful if the metadata has been manually moved

-n Use -n to force 'No' response to all queries. Default is to prompt.

-q Quiet. Displays minimal text output.

-xe *-xefile_name*

Redirects status, program, and error messages to *file_name*.

-xo *-xofile_name*

Redirects standard output to *file_name*.

-y Use -y to force 'Yes response to all queries. Default is to prompt.

-z Copies or moves a directory and all sub-directories. Requires the **-c**, **-d**, or **-r** option.

Special Considerations

- VTRANSFER resides on the client and references the archive path on the Version Manager File Server as viewed by the client.
- This command has no effect unless the archives are mapped to a Version Manager File Server.
- You can run the VTRANSFER command only if it is enabled by the **Enable Utilities** option in the Version Manager File Server Administration Utility.
- The VTRANSFER command is located in the bin\admin directory of your Version Manager installation. See your Version Manager administrator if you need to use this command but do not have access to it.

Examples

In the examples below, the archive is mapped to a Version Manager File Server using the client name S:\VMFS\PDBs.

Example 1: Export an Archive to a Zip File

```
vtransfer -x S:\VMFS\PDBs\PDB-1\archives\Project-1\Hello.txt-arc Hello.zip
```

Example-2: Export an Archive to a Zip File Using the Revision Path Stored in the Metadata

```
vtransfer -x -m S:\VMFS\PDBs\PDB-1\archives\Project-1\Hello.txt-arc Hello.zip
```

Example-3: Import an Archive from a Zip File

```
vtransfer -i S:\VMFS\PDBs\PDB-1\archives\Project-1\Hello.txt-arc Hello.zip
```

Example-4: Move an Archive

```
vtransfer -r S:\VMFS\PDBs\PDB-1\archives\Project-1\Hello.txt-arc
   S:\VMFS\PDBs\PDB-1\archives\Project-1\Boneyard\Hello.txt-arc
```

Example-5: Rename an Archive

```
vtransfer -r S:\VMFS\PDBs\PDB-1\archives\Project-1\Hello.txv
   S:\VMFS\PDBs\PDB-1\archives\Project-1\Hello.txt-arc
```

Example-6: Fix an Archive

```
vtransfer -f S:\VMFS\PDBs\PDB-1\archives\Project-1\Hello.txt-arc
```

Related Topics

For information about...	See...
Version Manager File Servers	*Administrator's Guide*
RFSSplitOnCreate directive	"RFSSplitOnCreate directive" on page 168
VSPLIT command	"VSPLIT command" on page 86

Chapter 2
Directives

Using Directives

Control aspects of program behavior

Directives are configuration options that you place in configuration files to define how Version Manager operates. The directives control the behavior of Version Manager commands. For example, the LogIn directive tells Version Manager which login sources to use to obtain user identification.

Configuration files enable you to provide the same set of options to all users, or different options for different sets of users, depending on the files they will access.

This section contains conceptual information about working with directives. This section contains the following topics about working with directives:

- Directive Types
- Master Configuration Files
- Configuration Files
- Using Conditional Constructs in Configuration Files

Directive Types

There are two types of directives: archive creation and operation:

- Archive creation
- Operation directives

The tables below describes the directive types. Each table contains a list of directive and its function.

Archive Creation Directives

Archive creation directives affect the attributes of archives created while these directives are in effect. *Changing an archive creation directive has no effect on existing archives*. To change the attributes of an existing archive, use the VCS command.

Use this directive...	To do this...
AccessList	Identify users and groups of users who are allowed to access archives.
CheckLock	Enable lock checking.
ColumnMask	Specify columns to ignore when comparing files.
CommentPrefix	Define the comment string that is inserted before lines in the Log keyword expansion.
Compress	Compress archives.
CompressDelta	Compress deltas only.
CompressWorkImage	Compress the workfile copies only.
ExclusiveLock	Prevent locking of more than one revision in an archive.
ExpandKeywords	Expand keywords when storing a revision.
GenerateDelta	Store each revision as a set of deltas.

Use this directive...	To do this...
NewLine	Change the end-of-line indicator in Log keyword expansion.
Owner	Assign ownership of archives.
RecordLength	Specify the logical record length.
Renumber	Renumber masked columns.
RFFSSplitOnCreate	Split archives into separate metadata and revision data if mapped to a Version Manager File Server.
Translate	Translate the end-of-line character.
WriteProtect	Write-protect archives.

Operation Directives

Operation directives take effect when you execute a command affected by the directives.

Use this directive...	To do this...
Abort	Stop all processing of configuration files.
AccessControl	Specify whether or not Version Manager uses an access control database.
AccessDB	Identify the access control database.
Alias	Refer to a character string symbolically.
ArchiveSuffix	Specify the extension used for archives.
ArchiveWork	Specify the directory for temporary archive copies.
AutoCreate	Create archives using the PUT command.
BaseVersion	Identify the revision where a branch begins.
BranchVersion	Identify the tip revision of a branch.
BranchWarn	Warn the user before creating a branch.
Case	Control the case-sensitivity of user IDs.
CtrlZ	Ignore characters after CTRL+Z.
DateFormat	Specify the format for date entry.
DefaultVersion	Identify the revision to use if no other is specified.
DeleteMessageFile	Delete message files after reading them.
DeleteWork	Delete workfiles after storing them.
DeltaDelete	Define the syntax for Insert commands in delta files.
DeltaInsert	Define the syntax for Delete commands in delta files.
DeltaReplace	Define the syntax for Replace commands in delta files.
DeltaSeq	Generate delta commands based on sequence numbers.
Diagnostic	Specify a diagnostic function.
Disallow	Prevent users from changing a directive.
Echo	Display a message.

Use this directive...	To do this...
End	Stop processing the configuration file containing this directive.
EndMaster	End the section of the file used as the master configuration file.
EventTrigger	Execute user-defined instructions before or after a Version Manager event.
ExpandKeywords Touch	Update the timestamp when expanding keywords.
FirstMatch	Control how wildcards are interpreted.
ForceUnlock	Remove the lock when storing an unchanged revision.
IgnorePath	Ignore the workfile path when locating archives.
Include	Read another configuration file.
Journal	Record an audit trail of archive modifications.
JournalReadOnlyGets	Record an audit trail of who got what files from source control.
LogIn	Specify the user identification source and default access control database privileges for each source.
MessageSuffix	Specify the extension of message files.
MonthNames	Specify different names for the months of the year.
MultiLock	Permit a user to lock more than one revision and/or permit more than one lock per revision.
PathSeparator	Specify the path separator character used in keyword expansion.
Promote	Define a promotion model.
Quiet	Display minimal messages.
ReferenceDir	Specify the directory where current workfile copies are automatically stored.
Semaphore	Guarantee exclusive access to shared archives.
SemaphoreDelay	Specify the delay between attempts to access an archive.
SemaphoreDir	Specify the directory where semaphore files are created.
SemaphoreRetry	Specify the number of attempts to access an archive.
SemSuffix	Specify the suffix used for semaphore files.
SignOn	Display the sign-on message.
VCSDir	Specify archive directories.
VCSEdit	Specify the text editor used for descriptions.
VCSID	Specify the ID of the current user.
WorkDir	Specify the directory for temporary files.
WriteProtect	Write-protect archives

NOTE To learn the syntax used to specify the above directives, see the individual directive topics in this manual or the *Command-Line Quick Reference*.

Directive rules

The following rules apply to directives:

- Directives are not case-sensitive.
- If a directive is defined more than once, Version Manager uses the last value defined.
- If the syntax for a directive includes an equals sign (=) between the directive and its parameter, you can substitute a space. Spaces before and after the equals sign are optional.

Default Directive Settings

Default settings for certain directives

There is no default configuration file associated with Version Manager in the command-line interface; there is, however, a default configuration file shipped with the product. This file is located in the lib directory within the Version Manager installation directory and is named default.cfg.

If you do not associate a configuration file with Version Manager when using the command-line interface, Version Manager uses the DTK defaults for the command-line interface, so operations performed using the desktop client may not have the same outcome as operations performed using the command line. See the *Administrator's Guide* for more information on default settings when using the desktop client and command-line interface.

Some directive attributes are set differently depending on the file type; others are applied universally to all file types. For example, the Translate directive is set differently for different types of files. The Translate directive could be used on text files, but setting this attribute for binary files may corrupt the archives.

If you do not associate a configuration file with Version Manager when using the command-line interface, Version Manager operates as follows:

- The archive suffix template is ??v___ . For example, a file named TEXT.TXT will be stored in an archive called TEXT.TXV. For more information, see the "ArchiveSuffix directive" on page 122.
- Archives are automatically created on check in if they do not exist. For more information, see the "AutoCreate directive" on page 126.
- Archives are write-protected. For more information, see the "WriteProtect directive" on page 179.
- The current working directory is where Version Manager looks for archives. For more information, see the "VCSDir directive" on page 175.
- Version Manager does not automatically store a reference copy of the workfile each time you check it into the archive. See the "ReferenceDir directive" on page 166.
- The user is not warned that a branch will be created. For more information, see the "BranchWarn directive" on page 127.
- Workfiles are deleted upon check in. For more information, see the "DeleteWork directive" on page 137.
- Keywords are not expanded, except for the following file types: .c, .h, .pas, .mak, .for, .bas (all platforms); .asm, .bat (Windows platforms). For more information, see the "ExpandKeywords directive" on page 149.
- End-of-line characters are not translated in all files, except for the following file types: .c, .h, .pas, .mak, .for, .bas (all platforms); .asm, .bat (Windows platforms). For more information, see the "Translate directive" on page 174.

- Version Manager uses a comment prefix of CommentPrefix .? = " * " before each line that results from the expansion of the Log keyword, except for the following file types, where the comment prefix is as follows:

All Platforms	Windows Only	UNIX Only
.c= " * "	.prg=" * "	.y= " * "
.h= " * "	.asm="; "	.l= " * "
.pas=" * "	.bat="rem "	.o= ""
.mak="# "	.obj=""	.a= ""
.for="C "	.lib=""	
.bas="rem "	.doc=""	

  For more information, see the "CommentPrefix directive" on page 130.

- Multiple locks on a single revision are not allowed. For more information, see the "MultiLock directive" on page 160.
- A user cannot lock more than one revision of an archive. For more information, see the "MultiLock directive" on page 160.
- More than one revision in an archive can be locked. For more information, see the "ExclusiveLock directive" on page 148.
- The semaphore suffix template is ??$___ , for example, the semaphore for the file TEXT.TXT would be TEXT.TX$. For more information, see the "SemSuffix directive" on page 173.
- The number of times Version Manager will attempt to access an archive is three. For more information, see the "SemaphoreRetry directive" on page 172.
- The delay between attempts to access an archive is one second. For more information, see the "SemaphoreDelay directive" on page 170.
- The login source is HOST. For more information, see the "LogIn directive" on page 156.
- A user's login ID is case sensitive. For more information, see the "Case directive" on page 128.
- The lock on an unchanged revision is not removed when check in is canceled because the file is unchanged. For more information, see the "ForceUnlock directive" on page 152.
- Archives are not compressed. For more information, see the "Compress directive" on page 131.
- Deltas are not compressed. For more information, see the "CompressDelta directive" on page 132.
- The initial copy of a file (work image) that is placed in an archive is not compressed. For more information, see the "CompressWorkImage directive" on page 133.
- Message files are not deleted after reading them. For more information, see the "DeleteMessageFile directive" on page 136.
- The default message suffix template is ??@___. For example, a workfile named TEST.C. would be assumed to have the message file name of TEST.C_@. For more information, see the "MessageSuffix directive" on page 159.

- Details of command operations are displayed. For more information, see the "Quiet directive" on page 164.
- A revision must be locked before you can check in a workfile. For more information, see the "CheckLock directive" on page 128.
- Revisions must be checked in by the user who locked the file. For more information, see the "CheckLock directive" on page 128.
- Revisions are stored as a set of deltas. For more information, see the "GenerateDelta directive" on page 152.
- The workfile path is used when locating archives. For more information, see the "IgnorePath directive" on page 153.
- The timestamp is updated when Version Manager expands keywords upon check in. For more information, see the "ExpandKeywords Touch directive" on page 150.
- Characters are not ignored after CTRL+Z. For more information, see the "CtrlZ directive" on page 133.

Master Configuration Files

Standardize configuration settings for all users

The master configuration file is a text file that contains directives and their settings. These directives define your configuration choices. Version Manager reads the *master configuration file* before any other configuration file. This ensures that all users share the same settings for certain directives. You can also use the master configuration file to prevent the use of certain directives in other configuration files so that users cannot use or reset them.

NOTE If there are blank spaces in the path to the configuration file, you must use quotation marks around the path for the command line interface. For example, if the sample project database is installed to the default location, define the path as:

```
include "C:\Users\All Users\Serena\VM\sampledb
\archives\basecfg.cfg"
```

Once you create a master configuration file, use the VCONFIG command to configure your Version Manager files to recognize it. Once embedded the file, Version Manager reads this configuration file first.

Two special directives

The following directives can only be used in the master configuration file:

- Disallow directive: Prohibits the use of certain directives in any other configuration file.
- EndMaster directive: When included in a file, this directive terminates its treatment as a master configuration file.

Examples

The following excerpt from a master configuration file ensures that all users follow the same archive suffix translation template:

```
ArchiveSuffix ??v__
Disallow ArchiveSuffix
```

These two lines can appear in either order before an EndMaster directive, because a disallowed directive applies only to subordinate configuration files, not to the master configuration file or to any of the files that it includes by means of the Include directive.

This example prevents users from redefining the LogIn directive to use a different source of user identification:

```
LogIn NetWare VLOGIN
Disallow LogIn
EndMaster
Include project.cfg
```

The last line of the previous example uses the Include directive to include the contents of PROJECT.CFG in the master configuration file. Because the EndMaster directive precedes this directive, Version Manager treats the contents of PROJECT.CFG as an ordinary configuration file.

NOTE

- It is recommended that you include the Semaphore, SemaphoreDir, ExpandKeywords Touch, and LogIn directives in the master configuration file. It is important that all users have the same settings for these directives.
- Make sure the master configuration file is only writable by the system administrator.
- To completely disallow a directive, you need to use Disallow on both the directive and the "No" form of the directive. If you disallow only the directive, subsequent configuration files can still specify the No form of the directive. For example, to prevent the use of MultiLock, specify:

  ```
  DISALLOW MultiLock NoMultiLock.
  ```

Related Topics

For information about...	See...
Preventing non-standard directive settings	"Disallow directive" on page 143
Specifying a file as the master configuration file	"VCONFIG command" on page 52
Including configuration files	"Include directive" on page 154
Preventing the use of directives	"Disallow directive" on page 143
Ending treatment of a file as the master configuration file	"EndMaster directive" on page 145
Keeping settings consistent for all users	"Semaphore directive" on page 169 "SemaphoreDir directive" on page 171 "ExpandKeywords Touch directive" on page 150 "LogIn directive" on page 156

Configuration Files

Control Version Manager commands

A *configuration file* is a text file that contains directives. You add and remove directives from the configuration file to control the way Version Manager operates.

Each Version Manager command looks for a configuration file first to determine the conditions under which to operate.

You can set up different configuration files for different users or projects. You can also use conditional constructs to direct Version Manager to read different parts of a configuration file under specific circumstances.

For more information on how to create and use configuration files, see the <Emphasis>PVCS Version Manager Administrator's Guide.

NOTE If there are blank spaces in the path to the configuration file, you must use quotation marks around the path for the command line interface. For example, if the sample project database is installed to the default location, define the path as:

```
include "C:\Users\All Users\Serena\VM\sampledb
\archives\basecfg.cfg"
```

How Version Manager Searches for Configuration Files

Where Version Manager checks for configuration files

Version Manager has a prescribed order in which it searches for configuration files. Version Manager first reads the configuration file that is embedded in the VCONFIG files (this is usually the master configuration file).

The most direct way to specify a configuration file is to use a command's -c option.

If you don't specify a configuration file using this option, Version Manager checks for a default configuration file (named vcs.cfg) in the current directory. If this file does not exist, Version Manager checks for the VCSCFG environment variable, which specifies the name and location of the configuration file to use.

If Version Manager can't find a master, project, or local (vcs.cfg) configuration file, it uses the default setting for each directive.

Suppress reading of a configuration file

If you use the -c option *without* specifying the name of a configuration file, Version Manager doesn't read any configuration file and bypasses reading the default configuration file.

To direct Version Manager to read both the default configuration file and another file for additional configuration information, use the -c- option. Then use the -c option to specify the additional configuration file.

The -c option should be the first option on the command line to prevent Version Manager from processing the command line with different directives than intended.

The manner in which Version Manager searches for configuration files means that each user can have a separate configuration file to tailor Version Manager to individual requirements. To prevent this individual customization and to and override certain Version Manager features you want to control, use the Disallow directive in the master configuration file.

Related Topics

For information about...	See...
Commands used in configuration files	"Using Directives" on page 105
Preventing non-standard directive settings	"Disallow directive" on page 143
Executing parts of a configuration file under different circumstances	"Using Conditional Constructs in Configuration Files" on page 113
Creating and using a master configuration file	"Master Configuration Files" on page 110

For more information on setting configuration options, see the <Emphasis>PVCS Version Manager Administrator's Guide.

Using Conditional Constructs in Configuration Files

Read directives conditionally

Use conditional constructs in configuration files to direct Version Manager to read different parts of the configuration file under specific circumstances.

For example, a system administrator may want to save time by including the configuration information for all projects in one configuration file—instead of providing one configuration file per project, which would require excessive maintenance.

The administrator instead creates the configuration file with conditional constructs and defines the directives for different projects.

Syntax

```
%if expression
    text
%elseif expression
    text
    .
    .
    .
%else
    text
%endif
```

The *expression* parameters can consist of conditional values or operators, used with string and numeric values. Version Manager resolves expressions to true or false to determine which lines of the file to read. Conditional construct parameters are not case sensitive.

The *text* that appears between clauses represents any valid configuration file text, including additional nested conditional constructs. The number of levels of nested constructs is limited by available memory.

Example

```
# If this file is being read by a command-line
# tool, include a command-line specific
# configuration file.
# The Command alias contains the name of the
# reading program (get, put, vcs, VMGUI)

%if "VMGUI" != "$(Command)"
include = \\server\project\cfg\master.cmdln
%endif
```

The master.cmdln might set the VCSDIR directive, as in the next example:

```
# Set archive dir dependent upon working dir

%if $(CWD) = "C:\work\project1"
VCSDIR="Z:\PROJECT1\ARCHS"
%elseif "$(CWD)" = "C:\work\project2" VCSDIR="Z:\PROJECT2\ARCHS"
%elseif "$(CWD)" = "C:\work\project3" VCSDIR="Z:\PROJECT3\ARCHS"
%endif
```

Version Manager has the following elements to aid in implementing conditional constructs:

- Conditional functions—built-in functions that execute conditional clauses.

- Operators—used to compare string and numeric values.
- String or numeric values—strings of characters or integers.

These elements are discussed in the following sections in greater detail.

Using Conditional Functions

Act on current conditions

Use the following built-in conditional functions to execute conditional clauses based on circumstances at the time the configuration file is read:

This conditional keyword...	Evaluates to...
%Exists (*file_name*) or %Exists(*directory*)	True if *file_name* or *directory* exists.
%File (*file_name*)	True if *file_name* exists.
%Dir (*directory*)	True if *directory* exists.
%Defined (*alias*)	True if *alias* is exists.

Using Operators

Operators are variables used in filter expressions; they compare string and numeric values. Version Manager evaluates operators with the same precedence level from left to right and evaluates negation from right to left. Use parentheses to clarify associations.

Compare values

Operators are listed below in decreasing order of precedence. Equivalent operators are grouped together.

Operator	Description
!	Logical negation (NOT)
<	Less than
>	Greater than
>=	Greater than or equal to
<=	Less than or equal to
== or =	Equal to
!= or <>	Not equal to
&& or &	Logical conjunction (AND)
^^ or ^	Logical exclusion (XOR)
|| or |	Logical disjunction (OR)

Using String or Numeric Values

Compare values with operators

Operators compare the following types of values:

- String values are strings of characters. Characters strings containing spaces or operators must be enclosed in double quotation marks ("). Strings may not contain double quotation marks.
- Numeric values are integers between 0 and 32,767.

In logical operations, Version Manager considers numeric values false if they are zero and true if they are other than zero. For string values, it considers a null string false and other string values true.

In relational operations where both operands are string values, Version Manager performs a case-sensitive comparison in which each character is weighted according to its ASCII value. When both operands are numeric values, Version Manager performs a comparison using 16-bit signed arithmetic (the total range of values is -32768 to +32767). If one of the operands is numeric and the other is a string, Version Manager converts the numeric value to its equivalent string and then performs a string comparison.

Example For each project, this configuration file uses a conditional construct to make project configuration almost completely automatic.

DOS
```
%if %defined(project)
     vcsdir = c:\serena\$(project)\c
     %if "$(vcsid)" == "millardf"
      %if %exists(c:\serena\$(project)\local.cfg)
       @c:\serena\$(project)\local.cfg
      %else
       @c:\serena\local.cfg
      %end
     %end
    %else
     vcsdir = c:\serena
    %end
```

UNIX
```
%if %defined(project)
     vcsdir = /usr/serena/$(project)/c
     %if "$(vcsid)" == "millardf"
      %if %exists(/usr/serena/$(project)/local.cfg)
       @/usr/serena/$(project)/local.cfg
      %else
       @/usr/serena/local.cfg
      %end
     %end
    %else
     vcsdir = /usr/serena
    %end
```

The outermost conditional construct checks for the existence of the alias *project*. If this alias is not defined, the *%else* clause sets VCSDir to C:\SERENA in Windows. For UNIX, the clause sets VCSDir to */usr/*serena.

If *project* is defined, another conditional construct tests the value of the alias VCSID. If it does not match the string *millardf*, processing ends. If it does match, a further test is made for the existence of a local configuration file. The outcome of this test determines which of two possible local configuration files is used.

NOTE

- To prevent Version Manager from expanding an alias to a null string, enclose it in double or single quotation marks (" or '). Otherwise, the null string equates to an expression with only one operand, which Version Manager cannot evaluate.
- Version Manager expands any aliases in an expression before it evaluates the expression.

Related Topics

For information about...	See...
Using configuration files	"Configuration Files" on page 111
Directives used in conditional constructs	"Abort directive" on page 116 "Echo directive" on page 143 "End directive" on page 144

The remainder of this chapter contains a detailed description of each directive in alphabetical order.

Abort directive

End processing

Use the Abort directive to terminate processing of the configuration file and return to the command line. Use this directive in conditional constructs to stop configuration file processing when Version Manager encounters a certain condition.

Directive type

Operation Directive. This directive takes effect when you use any Version Manager command.

Syntax

```
Abort [exit_code]
```

Enter an integer for the *exit_code* parameter to set the exit code.

Default

Not applicable

Example

In this configuration file excerpt, the Abort directive is used conditionally. If the user ID does not exist, configuration processing ends:

```
%if ("$(VCSID)" == "")
       echo *** You must set the VCSID environment variable ***
       abort 1
    %endif
    %if "$(mode)" == "test"
       end
    %endif
```

Special Consideration

Unlike the End directive, the Abort directive stops all configuration file processing, even if the file containing the directive was called from another file.

Related Topics

For information about...	See...
Exit codes	Your operating system manual
Another way to stop configuration file processing	"End directive" on page 144
Other directives	"Using Directives" on page 105

AccessControl directive

Control ACDB usage
Use the AccessControl directive to specify whether or not a project uses an access control database.

Directive type
Operation Directive.

Syntax
`[No]AccessControl`

Default
The default value is NoAccessControl.

Related Topics

For information about...	See...
Specifying an access database	"AccessDB directive" on page 117 "VCONFIG command" on page 52
Specifying default privileges	"LogIn directive" on page 156
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

AccessDB directive

Identify the access control database
Use the AccessDB directive to define the name and location of the access control database. The information you specify with this directive takes precedence over information specified using the command, VCONFIG -A.

Directive type
Operation Directive. This directive takes effect when you use the VCONFIG command.

Syntax
`AccessDB = file_name [[No]WriteProtect]`

The optional WriteProtect/NoWriteProtect keyword determines whether MAKEDB write-protects the access control database.

Default
When specifying an AccessDB, the default is WriteProtect. There is no default name or location for the access control database. You must specify the access control database name and location using either the AccessDB directive or the VCONFIG -A command.

Special Consideration
For maximum security, use the Disallow directive in the master configuration file to prevent the use of AccessDB in project configuration files.

Related Topics

For information about...	See...
Commands affected by this directive	"VCONFIG command" on page 52
Enabling and disabling access control	"AccessControl directive" on page 117
Preventing the use of the AccessDB directive	"Disallow directive" on page 143
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

AccessList directive

Identify users for access lists

Use the AccessList directive to specify the access list for new archives.

Directive type

Archive Creation Directive. The AccessList directive only affects archives when they are created. If you change this directive in your configuration file, it has no effect on existing archives. Use VCS -A to change the access list of an archive after it has been created.

Syntax

```
AccessList = [user_id[,user_id...]]
```

Use the *user_id* parameter to specify users or groups that are allowed access to the archive. Each user or group's level of access is determined by the privileges assigned in the access control database. Separate multiple user or group names with commas. If you omit the *user_id* parameter, new archives have empty access lists.

Default

No access list

Examples

- The following example places users on the access list:
  ```
  AccessList = pablop,eleanorr,lauriea
  ```
- The following example places users and groups on the access list:
  ```
  AccessList = gustavef,charlesd,engineering,qa
  ```

Special Considerations

- To use archive access lists, you must have set up an access control database.
- The AccessList specification is limited to 254 characters.
- The case of the user IDs or group names in the access list must match the case of the actual user IDs, unless you use the NoCase directive to make user IDs case-insensitive.

Related Topics

For information about...	See...
Making user IDs case-insensitive	"Case directive" on page 128
Changing the access list for existing archives	"VCS command" on page 56
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

AdvancedLabelFilter directive

Compress change information

Use the AdvancededLabelFilter directive to enable or disable usage of advanced label search options (regex is supported with Advanced label filtering) for vlog command.

Directive type

Operation Directive. This directive takes effect when you use the VLOG command.

Syntax

```
[No]AdvancedLabelFilter
```

Default

AdvancedLabelFilter is set by default.

Alias directive

Refer to text symbolically

Use the Alias directive to define an alias, which is a symbolic name that Version Manager uses to replace a character string that you have defined.

Directive type

Operation Directive. This directive takes effect when you use Version Manager commands.

Default

None

Syntax

```
Alias name = string_value
```

The *name* parameter can contain any alphanumeric characters and underscores (_). Other characters are not allowed.

Use the *string_value* parameter to specify the character string that you want Version Manager to replace with the alias *name*. To continue the string on succeeding lines, insert a backslash (\ ) at the end of each line.

Related Topics

For information about...	See...
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

Aliases

Represent character strings with one word

An *alias* is a symbolic name that Version Manager replaces with a character string that you define. You can use aliases to save keystrokes by defining a one-word alias to represent many words in command lines, configuration files, and list files. Aliases can represent a list of file names, a list of command-line options, a path name, or any combination of these character strings.

Defining Aliases

Three ways to define aliases

The following examples demonstrate the three methods of defining an alias.

Environment Variables: You can create an alias for a predefined environment variable:

Syntax

```
set alias_name=alias_value
```

Example

```
set input=keyboard.c keycodes.h serialp.c
```

Configuration File: You can define aliases in a configuration file using the Alias directive.

Syntax

```
Alias alias_name=alias_value
```

Example

```
Alias input=keyboard.c keycodes.h serialp.c
```

Command Line: You can define aliases on the Version Manager command line using the following syntax:

Syntax

```
command alias_name=alias_value [option...]
```

If you insert any spaces between the alias name, the equals sign, and the value, enclose the alias definition in double quotation marks (").

Example
```
get input="keyboard.c keycodes.h serialp.c"
```

Referencing Aliases

To reference aliases in commands, use the following syntax:

Syntax
```
command [option...] $(alias_name)
```

Enclose the alias name in parentheses and precede the first parenthesis with a dollar sign ($). Aliases are not case-sensitive.

You must define aliases before you reference them. Version Manager replaces undefined aliases with a null string without issuing a warning message.

Example If you define the alias input as the string keyboard.c keycodes.h serialp.c, the following command:
```
get -l $(input)
```

has the same effect as this command:
```
get -l keyboard.c keycodes.h serialp.c
```

If you reference an alias on the command line that is defined in a configuration file other than the default configuration file, you must use the -C option to specify that configuration file before referencing the alias.

Predefined Aliases

Aliases built into Version Manager

In addition to the aliases that you define, Version Manager includes several predefined aliases.

This alias...	Expands to...
Command	The name of the current Version Manager command.
CWD	The name of the current working directory.
PVCSVer	The version number of the current command.
LogInSource	The source of the user ID. Possible values are: HOST VCSID LANMAN VLOGIN NETWARE WNET UNKNOWN These values are the same as the parameters to the LogIn directive. The value UNKNOWN means that the program cannot derive a user ID from any source.
PvcsOSType	The operating system type Version Manager was built for. Possible values are: **PvcsOSType** — **Value** Windows — Windows Linux — Linux IBM RS-6000 — AIX Solaris — Solaris HP-UX — HP-UX

This alias...	Expands to...
PvcsOSArch	The operating system architecture Version Manager was built for. Possible values are:

PvcsOSType	Value
Windows	80x86 (32-bit)
	x86_64 (64-bit)
Linux	x86_64
IBM RS-6000	PPC64
Solaris	SPARCv9
HP-UX	IA64

System — The legacy name of the host operating system. Possible values are:

System	Value
Windows	Windows NT/80x86
Linux	Linux/80x86
IBM RS-6000	AIX/RS-6000
Sun	SPARC/Solaris
HP-UX	HP-UX/PA-RISC

As of Version Manager 8.6, the value returned by $(System) no longer matches the value shown on Command Line banners, which were changed as part of the switch to a 64-bit CPU architecture. So as not to break pre-8.6 configuration files, $(SYSTEM) now only returns legacy values.

VCSID — The current user ID. The value may be different at different points in the configuration file, especially before and after a LogIn directive.

Using Aliases

Reference groups of files

One use for aliases is to assign a name to groups of files that make up portions of a project. For example, you may want to divide a project into files that perform floating point or integer arithmetic. To maintain lists of these files, you could define two aliases in your configuration file:

```
alias float=procreal.c scient.c graph.c
    alias integer=procint.c doublpre.c bitfid.c
```

As long as you are using this configuration file, you can refer to these files using the alias in Version Manager commands:

```
get -l $(float)
    put -c- $(integer)
    vcs -v"relate completed" $(float) $(integer)
```

Special Considerations

- **For UNIX users:** Different UNIX systems and shells may handle environment variables differently. See your UNIX system documentation for more information.
- If you are using only the -#, -C, or @ options on the command line, precede the alias reference with the -C- option to direct Version Manager to read the configuration file at once, thereby ensuring that it recognizes the alias. If you're using any other command-line option, it's not necessary to use the -C- option.

- Aliases are similar in usage to Configuration Builder macros.

Related Topics

For information about...	See...
Using an alias in a configuration file	"AdvancedLabelFilter directive" on page 119
Using an alias to control conditional configuration file processing	"Using Conditional Constructs in Configuration Files" on page 113
Using an alias in a list file	"List and Group Files" on page 21

ArchiveSuffix directive

Specify an extension template

Use the ArchiveSuffix directive to specify the template Version Manager uses to compute an archive's extension (or suffix) from its workfile.

For more detail on how Version Manager uses the suffix template to translate the file extensions, see the Suffix Translation topic later in this section.

NOTE In previous versions of Version Manager, ArchiveSuffix was called LogSuffix. To maintain compatibility with configuration files, Version Manager still recognizes that name.

Directive type

Operation Directive. This directive takes effect when you use any Version Manager command. Archive Suffix statements can be given in any order. You may insert more than one ArchiveSuffix directive line in your configuration file and multiple file extensions are allowed. The effect is cumulative.

Syntax

```
ArchiveSuffix = suffix_template
```

The *suffix_template* can be a literal extension or a mask.

```
ArvhiveSuffix [=] .ext=[.]ext
ArchiveSuffix [=] .ext=[.]?v?___
```

The above archive suffixes override the default suffix for the specified extension. The first equals sign and the second period are optional.

Default

The default archive suffix template is ??v___, which generates archive names that you can refer to collectively using the wildcard file specification *.??v.

Examples

```
ArchiveSuffix = .frm=.vrm
ArchiveSuffix = .frx=.vrx
ArchiveSuffix = .frm .frx=v??
```

You can also specify extensions that you want to override the default suffix.

```
ArchiveSuffix  [=] .c .h .asm = [.]?v?___
```

Suffix Translation

Derive archive and message extensions

A *suffix template* is the pattern Version Manager uses to derive the extension (also called the suffix) of files it creates. Version Manager uses a suffix template based on the extension on the workfile to derive the archive extension and message file extension. And it uses a suffix template based on the extension of the archive to derive the semaphore file extension.

The following directives use suffix translation:

- ArchiveSuffix directive
- MessageSuffix directive
- SemSuffix directive

Suffix template format

A suffix template consists of six characters. The first three characters are the *suffix mask*, and the second three are the *suffix default*. These characters can be any valid file name characters, including question marks (?) as wildcard characters. If the suffix template contains spaces, enclose it in single or double quotation marks (' or ").

To derive an extension from the extension of another file (or *source extension*), Version Manager lays both the suffix mask and suffix default over the source extension. If the character in the suffix mask is other than ?, it uses that character. If the character in the source extension is present, it uses that character. Otherwise, it uses the suffix default character.

Examples

The following table shows examples of suffix translation:

Example	Workfile	Mask	Default	Archive
1	.INC	??V	___	.INV
2	.C	??V	___	.C_V
3	.INC	???	!!!	.INC
4	.C	???	!!!	.C!!
5	.INC	???		.INC
6	.C	???		.C
7	.INC	V??		.VNC
8	.C	V??		.V
9	.INC	V		.V
10	.C	V		.V
11	.INC	VCS		.VCS
12	.C	VCS		.VCS

In the table above, examples 5 through 12 show a null suffix default. Version Manager truncates the extension at the position of the first ?, for which there is no corresponding character in the source extension.

If there are source files with the same base file name but different extensions, such as FILE.C and FILE.H, examples 5 through 10 will result in file name conflicts and should not be used.

The suffix template used in examples 5 and 6 produces the same extension for the archive as for the workfile. So does the template used in examples 3 and 4, if the workfile extension is fewer than three characters long. If you use either of these templates, be sure to define the VCSDir directive so that archives are stored in a different directory than workfiles.

Special Considerations

- For file names that contain more than one period, the program considers that characters to the right of the rightmost period to be the extension.
- You can override the ArchiveSuffix directive by using the -s option with the following commands:

GET	PUT	VCS	VDEL
VDIFF	VLOG	VMRG	VSQL

- Version Manager supports file names longer than three characters. However, only the first three characters of the extension are affected by suffix translation.
- The ArchiveSuffix directive has an additional syntax for appending a string to file names to form archive names:
  `ArchiveSuffix +string`

  For example, if the directive is ArchiveSuffix +_v and the file name is *invent.c*, the resulting archive name would be *invent.c_v*.

Related Topics

For information about...	See...
Overriding the ArchiveSuffix directive on the command line	"GET command" on page 29 "PUT command" on page 38 "VCS command" on page 56 "VDEL command" on page 64 "VDIFF command" on page 67 "VLOG command" on page 76 "VMRG command" on page 81 "VSQL command" on page 88
Directives that use suffix translation	"ArchiveSuffix directive" on page 122 "MultiLock directive" on page 160 "SemSuffix directive" on page 173
Specifying archive directories	"VCSDir directive" on page 175
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

ArchiveWork directive

Work on archive copies

Use the ArchiveWork directive to tell Version Manager to make temporary copies of archives before updating them, and where to store them.

Version Manager copies the archive to a temporary file, modifies the copy, and then overwrites the original with your changes. This protects an archive from corruption should the system crash during the update process. Even if the original archive is corrupted, you can restore it from the temporary archive, which is still in the ArchiveWork directory under the name PV*nnnnnn*.TMP, where *nnnnnn* is a hexadecimal number.

Directive type

Operation Directive. This directive takes effect when you use the GET, PUT, VCS, VDEL, or VPROMOTE commands to update an archive.

Syntax

```
ArchiveWork = [path_name]
```

Default

The default ArchiveWork directory is the current working directory.

Special Considerations

- Don't define the ArchiveWork directive as a directory on a RAM disk. If the system crashes when Version Manager is copying the temporary archive over the previous archive, you would lose both the archive and the changed copy.
- In previous versions of Version Manager, the ArchiveWork directive was called LogWork. To maintain compatibility of configuration files, Version Manager still recognizes that name.
- Locate the ArchiveWork where access speed is reasonable, and where there is at least enough space to hold the largest archive.
- If you are using a Version Manager File Server, do not set ArchiveWork to a directory that is mapped to the file server.

Related Topics

For information about...	See...
Commands affected by this directive	"GET command" on page 29 "PUT command" on page 38 "VCS command" on page 56 "VDEL command" on page 64 "VPROMOTE command" on page 84
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

AutoCreate directive

Create archives automatically

Use the AutoCreate directive to cause the PUT command to create an archive automatically if it cannot find one for the workfile being checked in. Use NoAutoCreate to cause PUT to issue an error message instead of creating the archive.

Directive type

Operation Directive. This directive takes effect when you create an archive for a new workfile using the PUT command.

Syntax

```
[No]AutoCreate
```

Default

AutoCreate is the default.

Special Consideration

If AutoCreate is in effect and you change your PATH statement or the directories specified by the VCSDir directive, it may create a new archive in a different directory instead of storing the workfile in an existing archive.

Related Topics

For information about...	See...
Commands affected by this directive	"PUT command" on page 38
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

BaseVersion directive

Identify the revision where a branch begins

Use the BaseVersion directive to specify the version label that identifies the revision where a branch will begin. This revision is called the *branch point.*

Directive type

Operation Directive. This directive takes effect when you use any Version Manager primary command. It is used in conjunction with the BranchVersion and DefaultVersion directives.

Syntax

```
BaseVersion = version_label
```

Default

Not applicable

Related Topics

For information about...	See...
Referring to the version label on a branch	"BranchVersion directive" on page 127
Using directives	"Configuration Files" on page 111
Referring to the default version	"DefaultVersion directive" on page 135
Other directives including a list of primary directives	"Using Directives" on page 105

BranchVersion directive

Identify the branch tip revision

Use the BranchVersion directive to specify the version label assigned to a branch.

Directive type

Operation Directive. This directive takes effect when you use a Version Manager command.

Syntax

```
BranchVersion = version_label
```

Default

Not applicable

Related Topics

For information about...	See...
Assigning version labels	"VCS command" on page 56
Referring to the default version	"DefaultVersion directive" on page 135
Referring to the revision where a branch starts	"BaseVersion directive" on page 126
Commands affected by this directive	"Using Commands" on page 8
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

BranchWarn directive

Warn before creating a branch

Use the BranchWarn directive to force the GET -L and VCS -L commands to request confirmation before locking a revision that would result in a branch when you check in the workfile. If the CheckLock or ExclusiveLock directive is in effect, PUT warns you when you lock any non-tip revision. If MultiLock is in effect, PUT also warns you when you place a secondary lock on a revision.

If ExclusiveLock is in effect, the BranchWarn directive issues a warning when you lock a non-tip revision. This warning indicates that locking a non-tip revision automatically creates a branch when you check in the revision. With MultiLock, you see the warning when you lock any revision.

Directive type

Operation Directive. This directive takes effect when you use the GET -L or VCS -L command.

Syntax

```
[No]BranchWarn
```

Default

The default is NoBranchWarn.

Related Topics

For information about...	See...
Commands affected by this directive	"GET command" on page 29 "VCS command" on page 56
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

Case directive

Control user ID case-sensitivity

Use the Case directive to control the case-sensitivity of user IDs, regardless of their source.

Directive type

Operation Directive. This directive takes effect when you use any Version Manager command.

Syntax `[No]Case = VCSID`

The *VCSID* keyword is a required part of this directive, not a parameter that you replace.

Default By default, user IDs are case-sensitive.
Case=VCSID

Related Topics

For information about...	See...
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

CheckLock directive

Require locking in new archives

Use the CheckLock directive to ensure that users lock new archives. With CheckLock, Version Manager requires that:

- A revision must be locked before you can check in a workfile.
- Revisions must be checked in by the user who locked the file.

With NoCheckLock, Version Manager lets you check in a workfile without locking the revision.

NOTE If you try to lock or unlock a revision using the GET, PUT, or VCS commands, Version Manager issues an error message.

Directive type **Archive Creation Directive.** This directive only affects archives when they are created. If you change this directive in your configuration file, it has no effect on existing archives. After an archive has been created, use VCS +PL to enable lock checking and VCS -PL to disable lock checking.

Syntax `[No]CheckLock`

Default The default is `CheckLock`.

Related Topics

For information about...	See...
Changing lock checking for existing archives	"VCS command" on page 56
Other directives that affect lock checking	"ExclusiveLock directive" on page 148 "MultiLock directive" on page 160
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

ColumnMask directive

Exclude columns from comparison

Use the ColumnMask directive to specify the columns that Version Manager converts to spaces upon check in and the columns that should be treated as spaces when comparing or merging files. This feature is useful if you program in a language that uses line numbers, such as COBOL. If you don't use the ColumnMask directive, Version Manager assumes that every renumbered line has changed after you insert or delete lines.

Directive type

Archive Creation Directive. This directive only affects archives when they are created. If you change this directive in your configuration file, it has no effect on existing archives. Use the VCS -XColumnMask command to change column masking for existing archives.

Syntax

```
ColumnMask [.ext...]=col_start-col_end [(numeric)]...
```

Use the *col_start* and *col_end* parameters to specify column numbers. Column numbering begins with column number 1. You can define more than one *col_start-col_end* range. Ranges can overlap, and you can specify them in any order. If the columns to be masked are beyond the end of the record, the program does not pad lines with spaces.

Use the optional *.ext* parameter to apply this directive only to files with a certain extension. Enter a period (.) to apply the directive to files with no extension. Enter an asterisk (.*) to apply the directive to all files, regardless of their extension. Enter a question mark (.?) to apply the directive to files with extensions for which there is no defined or default value. Using "?" allows you to define a column mask for all extensions not otherwise defined.

Mask numeric fields only

To restrict masking to numeric fields only, use the keyword (Numeric) following a column range. In this case, the program only masks a field if the first character of the field is numeric.

Default

For COBOL files, use the following predefined value:

```
ColumnMask [.ext] = cobol
```

This is equivalent to the following:

```
ColumnMask [.ext] 1-6 (numeric), 73-80
```

Example

The following example masks columns 45 through 52, and masks columns 1 through 6 only if column 1 is numeric.

```
ColumnMask  1-6 (numeric), 45-52
```

Related Topics

For information about...	See...
Renumbering masked columns	"Renumber directive" on page 167
Overriding column masking	"GET command" on page 29
Changing column masking for existing archives	"VCS command" on page 56
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105
Keywords	*Administrator's Guide*

CommentPrefix directive

Define the comment identifier

Use the CommentPrefix directive to specify the characters that Version Manager inserts before each line that results from the expansion of the Log keyword.

Version Manager determines the comment prefix for certain types of files based on their extensions. Use the CommentPrefix directive for files whose extensions are not listed in the Default Comment Prefixes Table, or for which the default comment prefix is not appropriate.

Directive type

Archive Creation Directive. This directive affects archives only when they are created. If you change this directive in your configuration file, it has no effect on existing archives. Use the VCS -EC command to change the comment prefix of an archive after it has been created.

Syntax

`CommentPrefix [.ext...] [=] string`

Enclose the *string* parameter with double or single quotation marks (" or ') if it contains spaces, or if it begins with a period (.) and you don't supply the *.ext* parameter. Escape double or single quotation marks in *string* with a backslash (\).

Use the optional *.ext* parameter to apply this directive only to files with a specific extension.

- Enter a period (.) to apply the directive to files with no extension.
- Enter an asterisk (.*) to apply the directive to all files, regardless of their extension.
- Enter a question mark (.?) to apply the directive to files with extensions for which there is no defined or default value. Using "?" allows you to define a comment prefix for all "other" extensions not otherwise defined.

If you define more than one comment prefix for an extension, Version Manager uses the last one.

Default

CommentPrefix .? = " * ", except for the following file types, where the comment prefix is as follows:

All Platforms	Windows Only	UNIX Only
.c= " * "	.prg=" * "	.y= " * "
.h= " * "	.asm="; "	.l= " * "
.pas=" * "	.bat="rem "	.o= ""
.mak="# "	.obj=""	.a= ""
.for="C "	.lib=""	
.bas="rem "	.doc=""	

See "Default Directive Settings" on page 108 for more information.

Examples

The following examples demonstrate the use of *CommentPrefix*:

```
CommentPrefix.for c__
CommentPrefix.? ***
CommentPrefix.bas 'rem ** '
CommentPrefix.cbl '      *' for COBOL
```

Related Topics

For information about...	See...
Commands affected by this directive	"GET command" on page 29
Changing the comment prefix for existing archives	"VCS command" on page 56
Privileges that affect this directive	"ChangeCommentDelimiter privilege" on page 190
Escaping special characters	"Escape Characters in Specifications" on page 16
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105
Keywords	*Administrator's Guide*

Compress directive

Compress archive information

Use the Compress directive to enable or disable compression for new archives.

Directive type

Archive Creation Directive. This directive affects archives only when they are created. If you change this directive in your configuration file, it has no effect on existing archives. After an archive has been created, use VCS +PD or -PD to enable or disable delta compression, and VCS +PC or -PC to enable or disable text compression.

Syntax

```
[No]Compress [.ext...]
```

Use the optional *.ext* parameter to apply this directive only to files with a certain extension. Enter a period (.) to apply the directive to files with no extension. Enter an asterisk (.*) to apply the directive to all files, regardless of their extension. Enter a question mark (.?) to apply the directive to files with extensions for which there is no defined or default value.

Using the Compress directive turns on both CompressDelta and CompressWorkImage.

Default

The default is NoCompress.

Related Topics

For information about...	See...
Other directives that control compression	"CompressDelta directive" on page 132 "CtrlZ directive" on page 133
Enabling compression for existing archives	"VCS command" on page 56

For information about...	See...
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

CompressDelta directive

Compress change information

Use the CompressDelta directive to enable or disable compression for change information (or *deltas*) in new archives.

Directive type

Archive Creation Directive. This directive only affects archives when they are created. If you change this directive in your configuration file, it has no effect on existing archives. After an archive has been created, use VCS +PD to enable delta compression and VCS -PD to disable delta compression.

Syntax

`[No]CompressDelta [.ext...]`

Use the optional *.ext* parameter to apply this directive only to files with a certain extension. Enter a period (.) to apply the directive to files with no extension. Enter an asterisk (.*) to apply the directive to all files, regardless of their extension. Enter a question mark (.?) to apply the directive to files with extensions for which there is no defined or default value.

Default

The default is NoCompressDelta.

Related Topics

For information about...	See...
Other directives that control compression	"Compress directive" on page 131 "CtrlZ directive" on page 133
Enabling compression for existing archives	"VCS command" on page 56
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

CompressWorkImage directive

Compress workfile

Use the CompressWorkImage directive to enable or disable compression for the workfile copy (or *work image*) stored in new archives.

Directive type

Archive Creation Directive. This directive only affects archives when they are created. If you change this directive in your configuration file, it has no effect on existing archives. After an archive has been created, use VCS +PC to turn on workfile compression and VCS -PC to turn off workfile compression.

Syntax

`[No]CompressWorkImage [.ext...]`

Use the optional *.ext* parameter to apply this directive only to files with a certain extension. Enter a period (.) to apply the directive to files with no extension. Enter an asterisk (.*) to apply the directive to all files, regardless of their extension. Enter a

question mark (.?) to apply the directive to files with extensions for which there is no defined or default value.

Default — The default is NoCompressWorkImage.

Related Topics

For information about...	See...
Other directives that control compression	"Compress directive" on page 131 "CompressDelta directive" on page 132
Enabling compression for existing archives	"VCS command" on page 56
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

CtrlZ directive

Ignore characters after CTRL+Z — Use the CtrlZ directive to direct Version Manager to ignore any characters in a file after the CTRL+Z character.

Some programs use the CTRL+Z character as an end-of-file marker. In these files, any characters that appear after CTRL+Z are "garbage" characters and should be ignored.

Without the CtrlZ directive, Version Manager assigns no special meaning to CTRL+Z characters in workfiles.

When you use VDIFF to compare files with CtrlZ in effect, VDIFF stops comparing files when it encounters CTRL+Z.

Directive type — **Operation Directive.** This directive takes effect when you use the PUT command or VDIFF commands.

Syntax — [No]CtrlZ

Default — The default is NoCtrlZ.

Special Considerations

- Version Manager commands run slower when the CtrlZ directive is in effect because they have to check for CTRL+Z characters.
- Version Manager always treats CTRL+Z as the end-of-file character in configuration files, list files, and message files.
- **For UNIX users:** The CtrlZ directive has no effect on UNIX platforms.
- For binary files, set the following directives:
- NoExpandKeywords
- Notranslate
- NoCtrlz

Related Topics

For information about...	See...
Commands affected by this directive	"PUT command" on page 38 "VDIFF command" on page 67
Using directives	"Configuration Files" on page 111
Keywords	*Administrator's Guide*

DateFormat directive

Use a different input format for dates

Use the DateFormat directive to change the order of month, date, and year in dates that you enter, and to specify the characters used to separate dates and times. This directive only affects input format; it does not change how Version Manager displays dates and times.

Directive type

Operation Directive. This directive takes effect when you use the GET -D, GET -U, VJOURNAL -D, or VLOG -D command, or when you specify the date range during which a user ID is valid in the access control database.

Syntax

DateFormat = MM/dd/yyyy hh/mm/ss

The *MM*, *dd*, and *yyyy* parameters are literal text, not placeholders. You can specify them in any order. Notice the use of uppercase letters for the month. You will get the best results if you use this format. It is also recommended that you specify four digits for the year. You must specify the *hh*, *mm*, and *ss* parameters in the order shown.

Slashes (/) in the syntax statement above stand for the desired separator characters. In each position, specify one or more non-alphanumeric characters. There can be a different separator character in each position.

Default

The default date format is mmm/dd/yy hh:mm:ss, where *mmm* is the three-character English abbreviation for the name of the month, and times are on a 24-hour clock.

Example

The following example:

```
DateFormat = dd/mm/yy hh.mm.ss
```

Allows users enter dates in the following format:

```
07/02/99,3.45.22
```

NOTE You will get the best results if you specify the date format as dd/MM/yyyy or MM/dd/yyyy. Notice the use of uppercase letters for the month.

Special Consideration

You cannot use the same separator character for dates and times. A comma, space, or tab (as well as the last used separator) must be used to separate date fields from time fields. Note that a space cannot be used as a separator on the command line.

Related Topics

For information about...	See...
Changing the names of months	"MonthNames directive" on page 160
Commands affected by this directive	"GET command" on page 29 "VJOURNAL command" on page 73 "VLOG command" on page 76

DefaultVersion directive

Specify a default revision

Use the DefaultVersion directive to specify a version label or floating version label to be used in commands when no other revision or version is specified.

If DefaultVersion is not in effect and you don't specify a revision number in a command, Version Manager uses the latest revision on the trunk. You can use DefaultVersion to identify another revision for the program to use as the default. This way you don't need to manually specify the branch tip revision when you are working on a branch.

Directive type

Operation Directive. This directive takes effect when you use the commands in the following list:

GET	-L -P -R -T -V -W
PUT	-L -R
VCS	-L -R -V
VDEL	-R -V
VDIFF	-R -V
VLOG	-BN -BR -BV -R -V
VMRG	-R -V

You can override this directive on the command line by using a hyphen (-) after any of these options in place of the revision number or version label parameter.

Syntax

```
DefaultVersion = version_label
```

Default

TIP (most recent revision of the archive trunk)

Examples

- The following directive specifies *project1* as the default version:
  ```
  DefaultVersion = project1
  ```
  With the directive above in effect, the command checks out the revisions associated with the *project1* version label:
  ```
  get -l *.??v
  ```
- The following command checks out the tip revisions of the trunk instead of the version associated with the version label specified by the DefaultVersion directive:
  ```
  get -l- *.??v
  ```

Related Topics

For information about...	See...
Commands affected by this directive	"GET command" on page 29 "PUT command" on page 38 "VCS command" on page 56 "VDEL command" on page 64 "VDIFF command" on page 67 "VLOG command" on page 76 "VMRG command" on page 81
Using version labels	"Specifying Version Labels in Commands" on page 17
Referring to the revision where a branch starts	"BaseVersion directive" on page 126
Referring to the version label	"BranchWarn directive" on page 127
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

DeleteMessageFile directive

Delete message files after they are read

Use the DeleteMessageFile directive to delete message files after Version Manager reads them to obtain a change description or workfile description. For example, you might maintain a message file for a locked workfile while you are making changes to it. If you use this directive, you don't have to remember to delete the message file after it is used.

Directive type

Operation Directive. This directive takes effect when you use the PUT -M, PUT -T, VCS -M, or VCS -T command.

Syntax

```
[No]DeleteMessageFile
```

Default

The default is NoDeleteMessageFile.

Special Consideration

If you need to access the change description after a message file has been deleted, remember that the description is stored in the archive.

Related Topics

For information about...	See...
Commands affected by this directive	"PUT command" on page 38 "VCS command" on page 56
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

DeleteWork directive

Delete workfiles after storing

Use the DeleteWork directive to direct the PUT command to delete workfiles after check in.

If you use NoDeleteWork, Version Manager updates keywords in the workfile and leaves the workfile on disk. Specifying NoDeleteWork is equivalent to checking in a new revision and then immediately checking it back out without locking the revision.

Directive type

Operation Directive. This directive takes effect when you use the PUT command.

Syntax

```
DeleteWork
NoDeleteWork [No]WriteProtect
```

Use the optional [No]WriteProtect keyword to specify whether workfiles left on disk as a result of the NoDeleteWork directive are write-protected.

Use the NoWriteProtect keyword if other programs require your workfiles to be writable and you also want to enforce lock checking. (If you don't use lock checking, workfiles are always writable.)

Default

The default is DeleteWork. If you use NoDeleteWork, saved workfiles are write-protected by default.

Special Considerations

- Don't confuse the WriteProtect keyword with the WriteProtect directive, which determines whether archives are write-protected.
- If you use NoDeleteWork with NoWriteProtect, the GET command reminds you that a writable workfile exists and prompts you for permission to overwrite it.

Related Topics

For information about...	See...
Commands affected by this directive	"GET command" on page 29 "PUT command" on page 38
Lock checking	"CheckLock directive" on page 128
Write-protecting archives	"WriteProtect directive" on page 179
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105
Keywords	*Administrator's Guide*

DeltaDelete directive

Define syntax for Delete commands in delta files

Use the Delta Delete directive to define the syntax for Delete commands in delta files. The VDIFF -D command uses this syntax when generating a delta file. The delta file can then be used to generate a revision by applying the instructions in the delta file to a revision.

Directive type

Operation Directive. This directive takes effect when you use the VDIFF -D command.

Syntax `Delta Delete "format string"`

The *format* parameter describes the delta command. The *string* parameter can contain text and any of the following three variables:

%s	Start line (the line number of first line in a block)
%e	End line (the line number of last line in a block)
%c	Count (the number of lines in a block)

The first line in the reference file is line number 1. The Insert command uses %s to indicate the line *after* which text is inserted. For example, when inserting text between line 15 and 16, the value of %s is 15.

To accommodate delta formats where 0 is the first line in the file, or where the start line is used differently, you can use variables in simple arithmetic expressions, using this format: *variable*[+|-]*n*

For example, if line 0 is the first line in the file, use %s-1 and %e-1.

Examples

- In the example below, the Delete command has a hyphen (-) in front of it and a colon (:) after.
  ```
  Delta Delete "-delete:  %s, %c"
  ```
- In the example below, the Delete command is the same as the previous example, but it uses 0 as the first line.
  ```
  Delta Delete "-delete:  %s-1, %c"
  ```

Related Topics

For information about...	See...
Generating delta files	"VDIFF command" on page 67
Other directives that affect delta generation	"DeltaInsert directive" on page 138 "DeltaReplace directive" on page 139 "DeltaSeq directive" on page 140
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

DeltaInsert directive

Define syntax for Insert commands in delta files

Use the Delta Insert directive to define the syntax of Insert commands in a delta file. The VDIFF -D command uses this syntax when generating delta files. Then use the delta file to generate a revision by applying the instructions in the delta file to a revision.

Directive type

Operation Directive. This directive takes effect when you use the VDIFF -D command.

Syntax `Delta Insert "format string"`

The *format* parameter describes the delta command. The *string* parameter can contain text and any of the following three variables:

%s	Start line (the line number of first line in a block)
%e	End line (the line number of last line in a block)
%c	Count (the number of lines in a block)

The first line in the reference file is line number 1. The Insert command uses %s to indicate the line *after* which text is inserted. For example, when inserting text between line 15 and 16, the value of %s is 15.

To accommodate delta formats where 0 is the first line in the file, or where the start line is used differently, you can use variables in simple arithmetic expressions, using this format: *variable*[+|-]*n*

For example, if line 0 is the first line in the file, use %s-1 and %e-1.

Examples

- In the example below, the Insert command has a hyphen (-) in front of it and a colon (:) after.
  ```
  Delta Insert "-insert:  %s"
  ```
- In the example below, the Insert command is the same as the previous example, but it uses 0 as the first line:
  ```
  Delta Insert "-insert:  %s-1"
  ```

Related Topics

For information about...	See...
Generating delta files	"VDIFF command" on page 67
Other directives that affect delta generation	"DeltaDelete directive" on page 137 "DeltaReplace directive" on page 139 "DeltaSeq directive" on page 140 "GenerateDelta directive" on page 152
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

DeltaReplace directive

Define syntax for Replace commands in delta files

Use the Delta Replace directive to define the syntax for Replace commands in delta files. The VDIFF -D command uses this syntax when generating a delta file. Then use the delta file to generate a revision by applying the instructions in the delta file to a revision.

Directive type

Operation Directive. This directive takes effect when you use the VDIFF -D command.

Syntax `Delta Replace "format string"`

The *format* parameter describes the delta command. The *string* parameter can contain text and any of the following three variables:

%s	Start line (the line number of first line in a block)
%e	End line (the line number of last line in a block)
%c	Count (the number of lines in a block)

The first line in the reference file is line number 1. The Insert command uses %s to indicate the line *after* which text is inserted. For example, when inserting text between line 15 and 16, the value of %s is 15.

To accommodate delta formats where 0 is the first line in the file, or where the start line is used differently, you can use variables in simple arithmetic expressions, using this format: *variable*[+|-]*n*

For example, if line 0 is the first line in the file, use %s-1 and %e-1.

Examples

- In the example below, the Replace command has a hyphen (-) in front of it and a colon (:) after.
  ```
  Delta Replace "-replace: %s, %e"
  ```
- In the example below, the Replace command is the same as the previous example, but it uses 0 as the first line:
  ```
  Delta Replace "-replace: %s-1, %e-1"
  ```

Related Topics

For information about...	See...
Generating delta files	"VDIFF command" on page 67
Other directives that affect delta generation	"DeltaDelete directive" on page 137 "DeleteWork directive" on page 137
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

DeltaSeq directive

Generate delta commands based on sequence number

Use the Delta Seq directive to change the basis for delta file generation from line number to sequence number. Some programming languages, such as COBOL, let sequence numbers be placed in fixed locations in the file.

The VDIFF -D command uses the syntax defined by this directive when generating a delta file. You can then use the REGEN command to generate a revision by applying the instructions in the delta file to a revision.

Directive type **Operation Directive.** This directive takes effect when you use the VDIFF -D command.

Syntax `Delta Seq = start - end | COBOL`

Use the *start* parameter to specify the column where sequence numbers begin. Use the *end* parameter to specify the column where sequence numbers end. Values for *start* and *end* can be between 1 and 65535. The value for *end* must be greater than or equal to the value for *start*.

Use the alternative parameter COBOL to specify columns 1 through 6.

Example The following two directive lines are equivalent:

```
Delta Seq = 1 - 6
Delta Seq = COBOL
```

Related Topics

For information about...	See...
Generating delta files	"VDIFF command" on page 67
Other directives that affect delta file generation	"DeltaDelete directive" on page 137 "DeleteWork directive" on page 137 "DeltaReplace directive" on page 139 "GenerateDelta directive" on page 152
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

Diagnostic directive

Display diagnostic information Use the Diagnostic directive to display diagnostic information during command execution.

Directive type **Operation Directive.** This directive takes effect when you use any Version Manager primary or secondary command.

Syntax `Diagnostic = code`

Use the *code* parameter to select a diagnostic function. The possible values for *code* are:

0	Off.
1	Display each line of the configuration file as it is processed.
2	Display internal errors.
4	Toggle the sign-on message.
500	Display the archive owner, access list, access control database name, current user ID, and their privileges. The currently active privileges are in uppercase letters.

To specify more than one diagnostic item, add the diagnostic numbers. For example, use 3 to display the diagnostic information represented by numbers 1 and 2.

Default 0

Specify diagnostic functions from the command line

You can also specify the diagnostic number on the command line of any Version Manager command using the -# command-line option. Use the -# command-line option as the first option after the command name. Version Manager processes commands from left to right, and does not display diagnostic information until it reaches the -# option.

Related Topics

For information about...	See...
Commands that use the -# option	"GET command" on page 29 "MAKEDB command" on page 35 "PUT command" on page 38 "READDB command" on page 44 "REGEN command" on page 47 "VCOMPRES command" on page 50 "VCONFIG command" on page 52 "VCS command" on page 56 "VDEL command" on page 64 "VDIFF command" on page 67 "VJOURNAL command" on page 73 "VLOG command" on page 76 "VMRG command" on page 81 "VSQL command" on page 88
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

DisableBadLabelFilter directive

Assign labels that contain illegal characters

Use the DisableBadLabelFilter directive if you must assign new version labels that contain illegal characters. Else the filter will not allow you to create a label that contains illegal characters.

CAUTION! Disabling the check for illegal characters can result in unexpected and undesired outcomes.

In particular, the use of a dash (-) or plus sign (+) can cause an unwanted mathematical operation that results in Version Manager acting on a different revision than intended. For this reason, do **NOT** specify labels that end in a numeric value or the pound sign (#) preceded by a dash or plus sign. For example:

- abc-5
- abc+12
- abc-#
- abc+#

Directive type

Operation Directive. This directive takes effect when Version Manager encounters it in the configuration file for the project or project database.

Syntax

```
[No]DisableBadLabelFilter
```

Default `NoDisableBadLabelFilter`

Related Topics

For information about...	See...
Creating and using a master configuration file	"Master Configuration Files" on page 110
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

Disallow directive

Prevent the redefinition of directives

Use the Disallow directive in the master configuration file to prevent users from overriding certain directives. Version Manager disables the specified directives so that they have no effect when used in any other configuration file.

Directive type

Operation Directive. This directive takes effect when Version Manager encounters it in the master configuration file.

Syntax

`Disallow = directive...`

Specify the directives to disallow. Separate directives with spaces.

Default

Not applicable

Special Considerations

- You can only use the Disallow directive in the master configuration file.
- Even if you disallow a directive, you can still use it later in the master configuration file or in a configuration file included with the Include directive.
- A directive and its negative specification (e.g. MultiLock, NoMultiLock) must be disallowed separately. To disallow both at once, use Disallow MultiLock, NoMultiLock.
  `DISALLOW CTRLZ NOCTRLZ`

Related Topics

For information about...	See...
Creating and using a master configuration file	"Master Configuration Files" on page 110
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

Echo directive

Display a message

Use the Echo directive to display a message when Version Manager encounters this directive in a configuration file. This directive is useful for displaying status or debugging messages while a configuration file is being read.

Directive type **Operation Directive.** This directive takes effect as soon as Version Manager encounters it in a configuration file.

Syntax `Echo = message`

The *message* parameter can be any text. You do not need to enclose the text in quotation marks.

Default Not applicable.

Example In the following excerpt from a configuration file, the Echo directive is used to explain an error condition:

```
%if ("$(vcsid)" == "")
   Echo *** You must set the VCSID environment variable ***
   abort 1
%endif
%if "$(mode)" == "test"
   end
%endif
```

Related Topics

For information about...	See...
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

End directive

Stop configuration file processing Use the End directive to terminate processing the configuration file.

Directive type **Operation Directive.** This directive takes effect as soon as Version Manager encounters it in a configuration file.

Syntax `End`

Default Not applicable.

Example In the following configuration file excerpt, the End directive stops configuration file processing if the user ID does not exist:

```
%if ("$(vcsid)" == "")
  echo *** You must set the VCSID environment variable ***
  abort 1
%endif
%if "$(mode)" == "test"
  end
%endif
```

Special Consideration Unlike the Abort directive, if the configuration file was included by another configuration file, processing continues in the parent file.

Related Topics

For information about...	See...
Another directive that stops configuration file processing	"Abort directive" on page 116
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

EndMaster directive

End master configuration file processing

Use the EndMaster directive to stop treating the file that contains it as a master configuration file. This is useful if you want to include user-specific directives in the master configuration file.

Directive type

Operation Directive. This directive takes effect when Version Manager encounters it in a master configuration file.

Syntax

```
EndMaster
```

Default

Not applicable

Example

In the following example, the EndMaster directive indicates the end of master configuration file directives. The Include directive then reads in the contents of another configuration file.

```
Semaphore
Disallow Semaphore NoSemaphore Journal ArchiveWork
Disallow AutoCreate NoCheckLock NoExclusiveLock
EndMaster
Include project.cfg
```

Special Consideration

You can only use the EndMaster directive in the master configuration file.

Related Topics

For information about...	See...
Creating and using a master configuration file	"Master Configuration Files" on page 110
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

EventTrigger directive

Associate an event with an event name

Use the EventTrigger directive to identify specific events associated with an event name (see the following table). You can specify multiple EventTrigger directives, but only one event trigger is associated with any one event name.

For example, you may have separate event triggers for PrePut and PostPut, but you cannot have two different triggers both associated with PrePut. An event trigger specified for the event name, AllEvents, executes that trigger for every event. AllEvents is always called first.

See the <Emphasis>PVCS Version Manager Administrator's Guide for more information on event triggers.

NOTE Use the Disallow directive to disallow event triggers.

Directive type **Operation Directive.** This directive takes effect when Version Manager encounters it before an event in processing.

Exit Code EventTrigger returns an exit code of 0 if successful. A non-zero exit code terminates the event.

Default None

Syntax `EventTrigger [=] event_name [event_trigger_command]`

Version Manager supports the following values for the *event_name* parameter:

Use this event_name...	To execute a command...
AllEvents	Any event occurs. You can write a single event trigger to handle all events.
PostPut	After successfully creating a new revision.
PrePut	Just before checking in a workfile. The PrePut will occur only if there is nothing to prevent Version Manager from checking in the workfile.
UnconditionalPrePut	Before checking in a workfile; the workfile has not been read by Version Manager yet. Some errors may prevent the UnconditionalPrePut from occurring.
PreGet	Just before checking out a workfile.
PostGet	After successfully checking out a workfile.
PrePromote	Just before promoting a revision.
PostPromote	After successfully promoting a revision.
PreVersionLabel	Just before applying a version label.
PostVersionLabel	After applying a version label. PostVersionLabel will not occur when deleting a label.
PostJournal	Whenever an entry is made to the journal file (when the Journal directive is in effect).
PreLock	Just before locking a revision. This event occurs whether or not a revision is checked out at the same time.
PostLock	After locking a revision. This occurs whether or not a revision is checked out at the same time.
PreUnlock	Just before unlocking a revision. This event occurs when checking in an unchanged workfile if the ForceUnlock directive is in effect.

Use this event_name...	To execute a command...
PostUnlock	After unlocking a revision. This occurs when checking in an unchanged workfile, if the ForceUnlock directive is in effect.
PreCreateArchive	Just before creating a new archive. This occurs when checking in an initial revision if the AutoCreate directive is in effect.
PostCreateArchive	After creating a new archive. This occurs when checking in an initial revision if the AutoCreate directive is in effect.

Example

The event trigger PostPut C:\BIN\POSTPUT.BAT sends a message to a workgroup after Version Manager successfully checks in a revision:

```
PostPut.BAT :
      H:\PUBLIC\SEND "Just checked in %EventWorkfile%,
    revision %EventRevision%" to group SOFTDEV
```

Special Considerations

- It is recommended that you use the same name across platforms for event trigger executables. If you cannot do this, then address multiple platform differences with conditional constructs in conjunction with the system alias as shown in the following example.

  ```
  %if "$(SYSTEM)"== "HP-UX/PA-RISC"
  EventTrigger AllEvents eventHP.exe
  %else
  EventTrigger AllEvents eventwin.exe
  %endif
  ```

  As of Version Manager 8.6, the value returned by $(SYSTEM) no longer matches the value shown on Command Line banners, which were changed as part of the switch to a 64-bit CPU architecture. So as not break pre-8.6 configuration files, $(SYSTEM) now only returns legacy values. Had this not been done, the example above would have used the $(SYSTEM) value "HP-UX/IA64" resulting in eventwin.exe running on HP-UX. However, to access the real operating system type and CPU architecture of your Version Manager instance, use the aliases $(PVCSOSTYPE) and $(PVCSOSARCH). These aliases return "HP-UX" and "IA64" respectively for the example above.

  The example only needs to check for the OS type, however, it is forced to include the OS architecture as this is part of the $(SYSTEM) value. With the new alias, you can now use:

  ```
  %if "$(PVCSOSTYPE)" == "HP-UX"
  EventTrigger AllEvents eventHP.exe
  %else
  EventTrigger AllEvents eventwin.exe
  %endif
  ```

- Because Version Manager allows only one task or process at a time to access an archive, use caution when writing event triggers that attempt to modify the EventArchive. During Pre events, the archive is still open by the Version Manager process for which the event occurs. Semaphores prevent two processes from updating an archive at the same time, so you cannot update the archive from within a Pre event trigger. During Post events, the archive has already been closed, so it is safe to modify it.

- Using Configuration Builder as an event trigger, you can reference event information defined in the environment with $(EVENTARCHIVE) as long as you have not used the .NoEnvMacros directive in the build script. This Configuration Builder directive prevents environment variables from being defined as macros, so $(EVENTARCHIVE) would not be defined.
- Under Windows, the toolkit DLL can only be used by one program at a time. If your event trigger calls the Version Manager toolkit DLL directly, you must copy the DLL to a different name.

Related Topics

For information about...	See...
Using Configuration Builder	*Configuration Builder Reference Guide*
Specifying a directory for storing temporary files	"WorkDir directive" on page 179

ExclusiveLock directive

Prevent locking more than one revision of archive

Use the ExclusiveLock directive to prevent multiple locks in a single archive. Without ExclusiveLock, different users can place locks on different revisions in an archive at the same time.

Directive type

Archive Creation Directive. This directive only affects archives when they are created. If you change this directive in your configuration file, it has no effect on existing archives. After an archive has been created, use VCS +PE to enable exclusive locking and VCS -PE to disable exclusive locking.

Syntax

```
[No]ExclusiveLock
```

Default

NoExclusiveLock is the default.

Special Consideration

The ExclusiveLock and MultiLock directives are mutually exclusive. The MultiLock directive has no effect on an archive that was created while the ExclusiveLock directive was in effect. You must use the VCS -PE command to disable ExclusiveLock on the archive before you can have multiple locks in the archive.

Related Topics

For information about...	See...
Enabling exclusive locking for existing archives	"VCS command" on page 56
Other directives that affect revision locking	"CheckLock directive" on page 128 "MultiLock directive" on page 160
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

ExpandKeywords directive

Expand keywords during check in

Use the ExpandKeywords directive to enable or disable keyword expansion when revisions are checked in.

Directive type

Archive Creation Directive. This directive only affects archives when they are created. If you change this directive in your configuration file, it has no effect on existing archives. After an archive has been created, use VCS +PK to enable keyword expansion and VCS -PK to disable keyword expansion.

Syntax

```
[No]ExpandKeywords [.ext...]
```

Default

```
NoExpandKeywords
```

Use the optional *.ext* parameter to apply this directive only to files with a certain extension. Enter a period (.) to apply the directive to files with no extension. Enter an asterisk (.*) to apply the directive to all files, regardless of their extension. Enter a question mark (.?) to apply the directive to files with extensions for which there is no defined or default value.

IMPORTANT! Be certain to disable keyword expansion on any binary files that you place under Version Manager control since keyword expansion will change the length of the executable file without recalculating the offsets.

Examples

The following example turns off keyword expansion for all files and then turns it on for files with .C, .ASM, or .COB extensions. The UNIX example turns off keyword expansion for *.c*, *.s*, and *.pas* file extensions.

DOS

```
NoExpandKeywords
ExpandKeywords .c .asm .cob
```

UNIX

```
NoExpandKeywords
ExpandKeywords .c .s .pas
```

Related Topics

For information about...	See...
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105
Keywords	*Administrator's Guide*

Keywords

Keywords are special character strings that you can place in workfiles to define information about the archive when expanded. For example, the keyword $Archive$ defines the full path name of the archive. You can set a keyword option to tell Version Manager to expand keywords when the workfile is checked in.

The following table defines each Version Manager keyword. Keywords are case-sensitive.

Keyword	Definition
$Archive$	The full path name of the archive.
$Author$	The user ID of the revision author.
$Date$	The date the workfile was checked in.
$Header$	The archive name, revision number, revision date, and author ID.
Log	Cumulative check-in messages.
$Modtime$	The time of the last modification.
$Revision$	The revision number.
$Workfile$	The file name stored in the Version Manager archive.

CAUTION! Keyword expansion will cause corruption in binary files, so Version Manager disables keyword expansion by default, except for text files with certain file extensions. Do not enable keyword expansion for binary files.

Expanded keywords take the form $*Keyword:text*$, where text is the current value of the keyword. For example, the keyword $Revision$ could have the value of $Revision: 1.0$ when expanded. The Log keyword differs from the other keywords in that its expansion text is not enclosed by dollar signs. Instead, Version Manager inserts the change description after/in the line (depending on the value of new line character set for the given file type) that contains the Log keyword. The change descriptions accumulate in the workfile in reverse chronological order.

Fixed Length Keyword Expansion

Certain types of files must maintain a fixed line length in order to be read correctly. These types of files include column-sensitive languages, non-text files, word processor files, database files, and spreadsheet files. To maintain line length, you can regulate the exact length of the expanded keyword by placing character "fillers" between the keyword and its terminating dollar sign ($).

`$Header::xxx$`

Use the double colon (::) to select the fixed-length keyword expansion option. You can use any character as the x parameter. You can also vary the number of x parameters to determine the expanded length.

If the keyword expands to fewer than the specified number of characters, the program fills the remaining reserved space with spaces. If it expands to more than the required number of characters, the program includes as many characters as will fit.

ExpandKeywords Touch directive

Update time/date stamps when expanding keywords

Use the ExpandKeywords Touch directive to update the date and time stamps of workfiles when Version Manager expands keywords during check in. When you use Expandkeywords Touch to update workfiles, the *checked in* and *last modified* date and time attributes are modified to the current date and time.

If NoDeleteWork is set, this directive also updates the date and time of the workfile in your working directory.

Use the ExpandKeywords NoTouch directive to leave timestamps unchanged when keywords are expanded.

Directive type **Operation Directive.** This directive takes effect when you use the PUT command.

Syntax `ExpandKeywords [No]Touch`

Default ExpandKeywords NoTouch is the default.

Special Consideration If you use the ExpandKeywords Touch directive, it may affect processes that depend on accurate timestamps, such as builds.

Related Topics

For information about...	See...
Enabling keyword expansion	"ExpandKeywords directive" on page 149
Commands affected by this directive	"PUT command" on page 38
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105
Keywords	*Administrator's Guide*

FirstMatch directive

Change wildcard name search order Use the FirstMatch directive in conjunction with the VCSDir directive to modify the default method of expanding wildcards in archive specifications. When you use the FirstMatch directive, wildcard specifications match archives in the first VCSDir directory in which they are found.

The VCSDir directive identifies a list of directories in which archives are stored. If only VCSDir is in effect, Version Manager interprets wildcard specifications to match archives in all VCSDir directories.

Directive type **Operation Directive**. This directive takes effect when Version Manager expands wildcards.

Syntax `FirstMatch`

Example Assume that the following directive is in effect:

DOS `VCSDir = d:\project1;d:\project2`

UNIX `VCSDir = /project1;/project2`

If the FirstMatch directive is not set, the command get *.??v would check out all matching archives to D:\PROJECT1 and D:\PROJECT2 for Windows. For UNIX, the files would be

checked out to */project1* and */project2*. If the FirstMatch directive is set, the command only finds files in D:\PROJECT1 for Windows and in */project1* for UNIX.

Related Topics

For information about...	See...
How Version Manager expands wildcards	"Specifying Names Using Wildcards" on page 13
Specifying archive locations	"VCSDir directive" on page 175
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

ForceUnlock directive

Remove lock to store revision

Use the ForceUnlock directive to affect Version Manager's behavior when you choose not to store an unchanged revision. ForceUnlock directs PUT to remove the lock on the revision and delete the workfile (if DeleteWork is in effect), but not to increment the tip revision number. With NoForceUnlock, the PUT command is canceled, the lock remains in force, and the workfile is not affected.

ForceUnlock applies after you perform a put on an unchanged workfile and reply "n" when you get the message:

```
Workfile "..." unchanged, put anyway? (y/n)
```

Directive type

Operation Directive. This directive takes effect when you use the PUT command to store an unchanged, locked revision.

Syntax

```
[No]ForceUnlock
```

Default

NoForceUnlock is the default.

Related Topics

For information about...	See...
Commands affected by this directive	"PUT command" on page 38
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

GenerateDelta directive

Reduce computation time on revisions

The GenerateDelta directive controls how Version Manager stores revisions. As the default, it stores only the tip revision as a full copy of the workfile and all other revisions as a set of deltas. If workfiles are in binary, the set of deltas is often as large as the workfile, making delta computations very time-consuming. To save processing time, you can use the NoGenerateDelta directive to store complete copies of workfiles rather than a set of deltas containing only the differences.

Directive Type **Archive Creation Directive.** The GenerateDelta directive enables or disables delta generation for all archives or archives whose workfiles have specified extensions.

Syntax `[No]GenerateDelta [.ext ... ]`

Use the optional *.ext* parameter to apply this directive only to files with a certain extension. Use a period (.) to apply the directive to files without extensions. Use an asterisk (*) to apply the directive to all files, regardless of their extensions. Use a question mark (?) to apply the directive to files with extensions without default or defined values.

Default GenerateDelta is the default.

NOTE If you most commonly operate on trunk tip revisions, leaving the default in place is more efficient than using NoGenerateDelta.

After an archive has been created, use the VCS -PG and VCS +PG commands to enable or disable delta generation on new archives.

Related Topics

For information about...	See...
Creating new archives	"VCS command" on page 56

IgnorePath directive

Look for archives in VCSDir directories Use the IgnorePath directive to control how Version Manager searches for archives when you specify only a workfile name. When the IgnorePath directive is in effect, Version Manager ignores any path you specify for the workfile and looks in the VCSDir directories for the archive. If you don't specify a path for the workfile, the IgnorePath directive has no effect.

Use this directive to check out workfiles to directories other than the archive directories, and to make specifying workfile names more intuitive.

Directive type **Operation Directive.** This directive takes effect when Version Manager searches for an archive.

Syntax `[No]IgnorePath`

Default The default is NoIgnorePath.

Example This example assumes that the IgnorePath directive is in effect when you type the following command:

DOS
```
get d:\proj_1\input.c
```

UNIX
```
get /usr/serena/proj_1/input.c
```

The GET command initially ignores the specification D:\PROJ_1 for Windows and */usr/serena/proj_1* for UNIX, and attempts to locate the archive for INPUT.C in the VCSDir directories. If it finds the archive, it creates the workfile in the specified location.

Related Topics

For information about...	See...
How Version Manager searches for files	"How Version Manager Searches for Archives" on page 15
Specifying archive directories	"VCSDir directive" on page 175
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

Include directive

Read more configuration information

Use the Include directive to read an additional configuration file for supplemental configuration information. Configuration files can be nested up to 20 levels deep.

Syntax

`Include` *`configuration_file`*

Use the *`configuration_file`* parameter to specify the name of the configuration file. This can include a drive name and a path name. By default, Include looks in the current directory.

Default

None

Example

The following example instructs Version Manager to read the file MYVCS.CFG for additional configuration information:

```
Include myvcs.cfg
```

Special Consideration

The Include directive was formerly called the @ directive. Version Manager still recognizes it by that name.

Related Topics

For information about...	See...
Using directives	"Configuration Files" on page 111
Creating and using a master configuration file	"Master Configuration Files" on page 110
Other directives	"Using Directives" on page 105

Journal directive

Record all changes to archives

Use the Journal directive to instruct Version Manager to make an entry in the journal file after it modifies an archive. Using this directive provides you with an audit trail of changes to archives.

Directive type

Operation Directive. This directive takes effect when you use any Version Manager command.

Syntax `[No]Journal = [path_name][journal_file]`

The default name for the journal file is JOURNAL.VCS. You can keep a journal file for each archive and store it in the archive directory, or you can keep a journal file for all the archives in a project and specify its name and where to store it.

Default The default is NoJournal.

Examples

- This example names the journal file JOURNAL.VCS:
  `Journal`
- This example names the journal file PROJECT.JNL:
  `Journal = project.jnl`
- This example names the journal file JOURNAL.VCS and stores it in \PROJ_DIR\VCS_DIR for Windows. The UNIX example stores the journal file in */proj_dir/vcs_dir*:

DOS
```
Journal = \proj_dir\vcs_dir
```

UNIX
```
Journal = /proj_dir/vcs_dir
```

- This example names the journal file PROJECT.JNL and stores it in \PROJ_DIR\VCS_DIR for Windows. The UNIX example stores the file in */proj_dir/vcs_dir*:

DOS
```
Journal = \proj_dir\vcs_dir\project.jnl
```

UNIX
```
Journal = /proj_dir/vcs_dir/project.jnl
```

Related Topics

For information about...	See...
The contents of journal files	"LogIn directive" on page 156
Viewing journal files	"VJOURNAL command" on page 73
Using directives	"Using Directives" on page 105
Other directives	"JournalReadOnlyGets directive" on page 155

JournalReadOnlyGets directive

Record non-locking Get operations

Use the JournalReadOnlyGets directive to instruct Version Manager to make an entry in the journal file when ever a file is gotten from source control. Using this directive provides you with an audit trail of who got what files from source control.

NOTE This directive requires the journal directive. See "Journal directive" on page 154.

Directive type **Operation Directive.** This directive takes effect when you use the Version Manager Get command to get a file without locking the archive.

Syntax `[No]JournalReadOnlyGets`

Default The default is NoJournalReadOnlyGets.

Related Topics

For information about...	See...
Viewing journal files	"VJOURNAL command" on page 73
Using directives	"Using Directives" on page 105
Other directives	"Journal directive" on page 154

LogIn directive

Specify user identification sources

Use the LogIn directive to specify the sources Version Manager uses to obtain user identification. You can use this directive to specify the default privileges for each source.

When you use a Version Manager command, the program attempts to obtain user identification from each of the sources listed. If the program reaches the end of the list without obtaining user identification, it terminates and displays an access violation error message.

Directive type **Operation Directive.** This directive takes effect when you use any Version Manager primary or secondary command.

Syntax `LogIn source[(default privilege[,...])][,...]`

LogIn sources are not case-sensitive. Separate multiple sources with commas.

Default The default source for UNIX and Windows is HOST.

Source Values

Value	Meaning
Host ID	Host operating system. Use this source with systems that provide a user identification mechanism, such as UNIX, or for environments in which more than one network is in use. (Version Manager obtains the user ID from the operating system.) Use this option on UNIX and Windows systems where the user is required to enter a password. The directive for this option is LogIn=HOST.
LDAP ID	Lightweight Directory Access Protocol (LDAP). Use this source to authenticate user IDs and passwords against an LDAP server. Once authenticated against LDAP, user IDs are passed to the access control database, if one is in effect. The passwords, if any, in the access control database are ignored. The directive for this option is LogIn=LDAP. **NOTE** ■ LDAP does not work with the command-line interface (CLI). If LDAP is the first login source specified, the CLI will attempt to use the next login source. If no other login sources are specified, the CLI command will fail. ■ Microsoft Active Directory uses LDAP. ■ For clients that support dialogs, a Login dialog box appears if LDAP is set as your login source. Canceling, or failing, the login from this dialog box cancels the login operation. Additional login sources are NOT searched and you cannot log into the project database or project.
Login Dialog	The Version Manager desktop client login utility. This source requires users to enter a password before they can use Version Manager. To use password protection, an access control database must be defined. This source is only available in the desktop client. The directive for this option is LogIn=VLOGIN.
Netware ID	Novell NetWare (Windows only). Use this source to obtain user IDs from a Novell NetWare server, rather than Windows. The directive for this option is LogIn=NETWARE.

Value	Meaning
SSO/CAC	Single Sign On / Common Access Card. Use this source to authenticate user IDs and passwords or common access cards and PIN's against a Security Token Service (STS). The directive for this option is LogIn=SSO. SSO/CAC enables you to: ■ Login once to a given Version Manager client (Desktop, Web, Eclipse RIDE, and Visual Studio RIDE) and not have to login again during that client session, even when switching between project databases. ■ Login to SBM or the Version Manager Web client and not have to login to the other if you launch it as well. **NOTE** ■ SSO/CAC does not work with the command-line interface (CLI). If SSO/CAC is the first login source specified, the CLI will attempt to use the next login source. If no other login sources are specified, the CLI command will fail. ■ For clients that support dialogs, a Login dialog box appears if SSO/CAC is set as your login source. Canceling, or failing, the login from this dialog box cancels the login operation. Additional login sources are NOT searched and you cannot log into the project database or project. ■ To use SSO from the Version Manager Web client, you must configure both the project databases AND the servlets to use SSO. See the *Installation Guide*. ■ This login source requires a Single Sign On server to authenticate against. See the *Installation Guide*. ■ No other login source should appear before SSO/CAC . ■ Any login sources below SSO/CAC will apply only to CLI.
VCS ID	The user's Version Manager ID, which Version Manager derives from the value of the VCSID environment variable. The directive for this option is LogIn=VCSID. Be aware that using VCSID as a source for user identification is not secure. Users can circumvent security by logging in as another user or resetting the value of the VCSID environment variable.
Wnet ID	Microsoft Windows networks. Version Manager obtains the user ID from the Microsoft WNET API. The directive for this option is LogIn=WNET.

Version Manager checks the sources in the *source* parameter in order from left to right, and uses the first one that produces a user ID. If you omit the *source* parameter, a null value overrides any valid sources you may have specified previously.

Special Considerations

- The LogIn source VCSID is not a secure source of user identification since users can create local configuration files or redefine the VCSID environment variable.
- For information on automatically creating users upon login, see the *Administrator's Guide*.
- If you are using the IBM LAN Server, Microsoft LAN Manager or Novell NetWare networks, you should also use the Network directive in your configuration file.

- The LogIn directive overrides the user ID sources specified by the VCONFIG -I command, unless otherwise restricted. For maximum security, it is suggested that you disallow the LogIn directive.
- **For UNIX users:** When used from the CLI, the UNIX version of Version Manager supports only the *HOST* and *VCSID* options. The default value is *HOST*.

Related Topics

For information about...	See...
Another way to specify the user identification source	"VCONFIG command" on page 52
Enabling/disabling access databases	"AccessControl directive" on page 117
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105
Privileges	"Using Privileges" on page 183

MessageSuffix directive

Specify message file suffix

Use the MessageSuffix directive to specify the template that Version Manager uses to translate workfile extensions into message file extensions.

A *message file* contains text to be used as a change or workfile description.

The following commands specify message files:

PUT -M@
PUT -T@
VCS -M@
VCS -T@

Directive type

Operation Directive. This directive takes effect when you use the PUT -M, PUT -T, VCS -M, or VCS -T command.

Syntax

`MessageSuffix = suffix_template`

Default

The default suffix template is ??@___, which produces the message file name of TEST.C_@ for a workfile named TEST.C.

MessageSuffix=??@___

Related Topics

For information about...	See...
Specifying suffix templates	"Suffix Translation" on page 123
Commands affected by this directive	"PUT command" on page 38 "VCS command" on page 56
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

MonthNames directive

Specify different names for months

Use the MonthNames directive to specify different names for the months of the year, as well as for "AM" and "PM."

You can use the new month names when entering any command that accepts a date as a parameter. Version Manager also uses the different month names wherever it displays or prints dates.

Directive type

Operation Directive. This directive takes effect when you use any Version Manager command that accepts dates as input.

Syntax

```
MonthNames = m1 m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 am pm
```

Use the *m1* through *m12* parameters to specify the month names in order. Use the *am* and *pm* parameters to specify the identifiers for times that are after or before midnight. You cannot use numbers or symbols in these parameters.

Default

Version Manager uses English language names for months unless you use this directive to change them.

Example

The following example specifies French names for months:

```
MonthNames = janvier février mars avril mai \
    juin juillet août septembre octobre \
    novembre décembre AM PM
```

Special Considerations

- When it displays or prints month names, Version Manager uses the capitalization specified in the MonthNames directive.
- Month names and AM/PM indicators must be unique.

Related Topics

For information about...	See...
Specifying the date input format	"DateFormat directive" on page 134
Commands affected by this directive	"GET command" on page 29 "VJOURNAL command" on page 73 "VLOG command" on page 76
Specifying dates and times	"Specifying Dates and Times" on page 23

MultiLock directive

Maintain multiple locks in an archive

Use the MultiLock directive to permit users to lock more than one revision and/or to permit more than one lock per revision.

When the MultiLock directive is in effect and you direct Version Manager to lock a revision that is already locked, Version Manager displays a message indicating the revision is already locked. Then it prompts you to confirm the second lock.

Directive type

Operation Directive. This directive takes effect when you use any Version Manager command that involves locking revisions.

Syntax `[No]MultiLock [revision|user]`

You can use the revision or user keyword to restrict the type of multiple locking that is permitted. If you omit both the revision and user keywords, a user can place multiple locks on any revision in an archive.

Default

The default is NoMultiLock. Multiple users can place hold locks on different revisions, but not on the same revision.

Two forms of the directive

- **MultiLock Revision** lets multiple users lock the same revision in an archive, but no one user can lock more than one revision in an archive.

  When the user with the first lock checks in a revision, Version Manager creates a new revision on the same development path (trunk or branch). When the user with any subsequent lock checks in a revision, Version Manager checks it in as a new branch.

- **MultiLock User** lets a user lock multiple revisions in an archive, but no revision can have more than one lock.

  If you have two locks in the same archive, and are ready to check one in, you must specify one of the revision numbers that was checked out, and Version Manager will select the next available higher number from the specified sequence. Otherwise, you can specify the revision number.

Special Considerations

- To ensure that all users use MultiLock consistently, put the MultiLock directive in the master configuration file and then disallow MultiLock and ExclusiveLock.
- The ExclusiveLock and MultiLock directives are mutually exclusive. The MultiLock directive has no effect on an archive that was created while the ExclusiveLock directive was in effect. You must use the VCS -PE command to disable ExclusiveLock on the archive before you can have multiple locks in the archive.
- ExclusiveLock limits an archive to one lock.

Related Topics

For information about...	See...
Preventing multiple locks on an archive	"ExclusiveLock directive" on page 148
Commands affected by directive	"PUT command" on page 38 "VCS command" on page 56
Creating a master configuration file	"Master Configuration Files" on page 110
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

NewLine directive

Change end-of-line indicators in keyword expansion

Use the NewLine directive to specify the character string used to terminate lines added to a workfile when expanding the Log keyword.

If you do not specify this directive, the program uses a carriage return/line feed combination. Specific operating systems and environments may require a different end-of-line marker.

Directive type **Archive Creation Directive.** This directive only affects keyword expansion in archives when they are created. If you change this directive in your configuration file, it does not affect keyword expansion in existing archives. Use the VCS -EN command to change the end-of-line string after an archive has been created.

Syntax `NewLine [.ext...] = string`

Use the following symbols to indicate special characters in the *string* parameter:

\rCarriage return
\nLine feed
\tTab

You can place any character in the *string* parameter using the form \0X*nn*, where *nn* is the hexadecimal equivalent of the desired character. Version Manager interprets the character \0X00 as the end of the string.

Use the optional *.ext* parameter to apply this directive only to workfiles with a certain extension. Enter a period (.) to apply the directive to workfiles with no extension. Enter an asterisk (.*) to apply the directive to all workfiles, regardless of their extension. Enter a question mark (.?) to apply the directive to workfiles with extensions for which there is no defined or default value.

Default If you do not specify this directive, the program uses a carriage return/line feed combination (\r\n). Specific operating systems and environments may require a different end-of-line marker.

Related Topics

For information about...	See...
Commands affected by this directive	"PUT command" on page 38
Using keyword expansion	"PUT command" on page 38
Changing the end-of-line string for existing archives	"VCS command" on page 56
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105
Keywords	*Administrator's Guide*

Owner directive

Use the Owner directive to specify the user ID of the owner of new archives. If you do not use the Owner directive, new archives are owned by the user who creates them.

Directive type **Archive Creation Directive.** This directive only affects archives when they are created. If you change this directive in your configuration file, it has no effect on existing archives. After an archive has been created, use VCS -O to change the archive owner.

Syntax `Owner = user_id`

Use the user ID to specify the owner of all new archives.

Default Archives are by default owned by the user who creates them.

Special Consideration

The owner of an archive has no special access privileges to the archive. The owner must be included in the access list and have adequate privileges to access the archive, just like any other user.

Related Topics

For information about...	See...
Changing the owner of existing archives	"VCS command" on page 56
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

PathSeparator directive

Change the component delineator in path names

Use the PathSeparator directive to change the character Version Manager uses to separate the parts of path specifications that appear in expanded keywords such as $Archive$.

The Version Manager default is to use forward slashes (/) to separate the path components of file names in expanded keywords since most of the popular languages, such as C, use the backslash as an escape character.

If you plan to embed keywords in compiled files under UNIX platforms, or to use the correct path separator under PC platforms, use the PathSeparator directive to set the path separator to \.

Directive type

Operation Directive. This directive takes effect when the PUT command expands keywords in a file.

Syntax

```
PathSeparator = \ | /
```

The only allowable parameters for the PathSeparator directive are the forward slash (/) used in UNIX platforms, and backslash (\) used in PC platforms, followed by an optional space.

Default

The default character is a forward slash (/).

Special Considerations

- You can use the backslash (\) as a line continuation character as well as a path separator. To ensure that Version Manager does not mistake the path separator definition for a line continuation character, put a space after the backslash.
- Certain text editors strip spaces from the ends of lines when saving a file. If you use one of these editors, put a space and then any other character at the end of your PathSeparator definition.
- **For UNIX users:** The PathSeparator directive has no effect under UNIX.

Related Topics

For information about...	See...
Commands affected by this directive	"PUT command" on page 38
Using keyword expansion	"PUT command" on page 38
Using directives	"Configuration Files" on page 111

For information about...	See...
Other directives	"Using Directives" on page 105
Keywords	*Administrator's Guide*

Promote directive

Identify promotion groups

Use the Promote directive to specify a promotion model for your project by defining each promotion group in relation to the next higher group.

Directive type

Operation Directive. This directive takes effect when you use the GET, PUT, VCS, VLOG, or VPROMOTE commands.

Syntax

```
Promote from_group to_group
```

Default

None

Examples

The following examples define a promotion hierarchy that progresses in the following order: development, quality assurance (QA), beta release, and general availability (GA).

```
Promote development QA
    Promote QA beta
    Promote beta GA
```

Special Considerations

- Promote directives can appear in any order in the configuration file.
- Promotion group names are case-sensitive.

Related Topics

For information about...	See...
Promoting revisions	"VPROMOTE command" on page 84
Commands affected by this directive	"GET command" on page 29 "PUT command" on page 38 "VCS command" on page 56 "VLOG command" on page 76 "VPROMOTE command" on page 84
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

Quiet directive

Minimize screen messages

Use the Quiet directive to direct Version Manager to display only minimal messages.

Directive type

Operation Directive. This directive takes effect when you use any Version Manager primary or secondary command.

Syntax `Quiet | Verbose`

With the Verbose directive in effect, Version Manager displays details of command operation such as the workfile and archive names and the current and previous revision numbers.

Default The default is Verbose.

Related Topics

For information about...	See...
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

RecordLength directive

Specify logical record length

Use the RecordLength directive to specify the record length of fixed-length files so Version Manager can generate deltas based on logical lines. Using this directive improves Version Manager's performance when dealing with fixed-length records.

If you don't use the RecordLength directive, Version Manager generates deltas that are larger then they would be otherwise. Also, when you use VDIFF to display the differences between two revisions, the output is more difficult to decipher.

Variable-length vs. fixed-length records

Most DOS text files have variable-length records. A *newline sequence* (carriage return/line feed) marks the end of each line. Most text files created on mainframe operating systems, such as MVS, do not use this end-of-line sequence. Every record has the same number of characters. This number is called the *logical record length*. Files of this type are known as *fixed-length files*.

When fixed-length files are transferred from a mainframe to a DOS system, they are usually translated from fixed-length to variable-length by adding a newline sequence after each logical record, and sometimes by stripping trailing blanks.

Version Manager treats fixed-length files similarly to binary files. Tabs and newline characters have no special meaning.

Directive type **Archive Creation Directive.** This directive only affects archives when they are created. If you change this directive in your configuration file, it has no effect on existing archives. Use VCS -XRecordLength to change the record length for existing archives.

Syntax `RecordLength [.ext...] = record_length`

Enter the number of characters in a line for the *record_length* parameter. The maximum record length is 64K. To turn off the record length, use zero.

Use the optional *.ext* parameter to apply this directive only to files with a certain extension. Enter a period (.) to apply the directive to files with no extension. Enter an asterisk (*) to apply the directive to all files, regardless of their extension. Enter a question mark (?) to apply the directive to files with extensions for which there is no defined or default value.

Special Considerations

- If you use keywords in fixed-length record files, they must be in fixed-length format. As with binary files, using variable-length keywords can produce unexpected results.

- Do not use the RecordLength directive when you create an archive for a variable length file. If you do, this can potentially corrupt an archive or cause other unexpected results.

Related Topics

For information about...	See...
Fixed-length keywords	"PUT command" on page 38
Changing the record length for existing archives	"VCS command" on page 56
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105
Keywords	*Administrator's Guide*

ReferenceDir directive

Specify directory for copies of revisions

Use the ReferenceDir directive to specify location where Version Manager automatically stores a copy of each workfile each time it is checked into an archive.

Directive type

Operation Directive. This directive takes effect when you use the PUT command to create a new revision.

Syntax

```
ReferenceDir [=] [keyword ...] [directory]
```

Default

Version Manager does not automatically store a copy of the workfile each time you check it into the archive.
ReferenceDir=

Using the keyword parameter

Use the *keyword* parameter to control the use and attributes of the files in the reference directory. Separate multiple keywords with spaces. ReferenceDir keywords are not case-sensitive. If you use mutually exclusive keywords, Version Manager uses the last value.

Keyword Values

TrunkOnly	Use the reference directory only for revisions added to the trunk. This is the default.
All	Use the reference directory for trunk and branch revisions.
WriteProtect	Write-protect the files in the reference directory.
NoWriteProtect	Leave the files in the reference directory writable. This is the default.

Specifying directory paths

The *directory* parameter can be an explicit path name, a path name relative to the current directory, or a path name relative to the archive directory.

To create the reference directory relative to the archive directory, begin the directory path name with an asterisk (*). To specify the beginning of a reference directory hierarchy, end the directory path name with an asterisk.

If you don't specify the *directory* parameter, Version Manager creates reference files in the same directory as the archives.

Examples

- The following example creates a reference directory using an absolute path name:

DOS

```
ReferenceDir c:\serena\ref_dir
```

UNIX

```
ReferenceDir /usr/serena/ref_dir
```

- The following example creates a reference directory relative to the current directory:

DOS

```
ReferenceDir ..\ref_dir
```

UNIX

```
ReferenceDir ../ref_dir
```

- The following example creates a reference directory as a subdirectory of the archive directory, which is denoted by the asterisk:

DOS

```
ReferenceDir *\ref_dir
```

UNIX

```
ReferenceDir */ref_dir
```

- The following example creates a reference directory structure that mirrors the archive directory structure:

DOS

```
ReferenceDir "C:\Serena\vm\ref_dir\*"
vcsdir "C:\Serena\vm\archdir\*"
```

UNIX

```
ReferenceDir "/usr/serena/vm/ref_dir/*"
vcsdir "/usr/serena/vm/archdir/*"
```

Special Considerations

- Any keyword you use with ReferenceDir applies to all reference directories specified with any ReferenceDir or VCSDir statement
- If you do not specify the *directory* parameter but specify a keyword, then the keyword applies to all directories specified as reference directories using the VCSDir directive.

Related Topics

For information about...	See...
Commands affected by this directive	"PUT command" on page 38
An alternate way to specify reference directories	"VCSDir directive" on page 175
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

Renumber directive

Renumber masked columns

Use the Renumber directive to specify the columns in which Version Manager inserts line numbers when retrieving a workfile.

Use this feature in conjunction with the ColumnMask directive. Version Manager only renumbers masked columns. For example, you would mask COBOL files in columns 1-6 and 73-80, and renumber them in columns 1-6.

Directive type

Archive Creation Directive. This directive only affects archives when they are created. If you change this directive in your configuration file, it has no effect on existing archives. To change renumbering for existing archives, use the VCS -XRenumber command.

Syntax

```
Renumber [.ext...]=col_start-col_end [from start by number]
```

Use the *col_start* and *col_end* parameters to specify column numbers. Column numbering begins with column number 1. *Start* is the number to begin numbering with and *number* is the amount by which to increase this number for each line. Version Manager fills line numbers with zeroes beginning on the left (000010, 000020, etc.)

Use the optional *.ext* parameter to apply this directive only to files with a certain extension. Enter a period (.) to apply the directive to files with no extension. Enter an asterisk (.*) to apply the directive to all files, regardless of their extension. Enter a question mark (.?) to apply the directive to files with extensions for which there is no defined or default value.

Default

Do not renumber (based on file extension).

For COBOL files, use the following predefined value:

```
Renumber [.ext...] = cobol
```

This is equivalent to the following:

```
Renumber 1-6 from 10 by 10
```

Example

Using COBOL renumbering, a file would appear as follows:

```
000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID.NWPR0001.
000030 DATE-COMPILED.01/24/91
000040 ENVIRONMENT DIVISION.
```

Related Topics

For information about...	See...
Masking columns	"ColumnMask directive" on page 129
Specifying the record length of fixed-length files	"RecordLength directive" on page 165
Changing renumbering for existing archives	"VCS command" on page 56 "GET command" on page 29
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

RFSSplitOnCreate directive

Separate revisions and metadata

Use the RFSSplitOnCreate directive to split new archives into separate revision and metadata stores for use with the revision library feature of the Version Manager File Server.

Use this feature in conjunction with a Version Manager File Server. By default, it is enabled in newly created project databases.

Directive type **Archive Creation Directive.** This directive affects archives only when they are created. If you change this directive in your configuration file, it has no effect on existing archives. To split existing archives, use the VSPLIT command.

Syntax `[No]RFSSplitOnCreate`

Default NoRFSSplitOnCreate is the default.

Special Consideration This directive has no effect unless the archives are mapped to a Version Manager File Server.

Related Topics

For information about...	See...
Version Manager File Servers	*Administrator's Guide*
The VSPLIT command	"VSPLIT command" on page 86 *Administrator's Guide*
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

Semaphore directive

Guarantee exclusive access to shared archives Use the Semaphore directive to control whether Version Manager uses semaphores to guarantee exclusive access to archives, journal files, and the access control database, and what type of semaphores it uses.

Use this directive in network environments, where archives are shared by multiple users.

Directive type **Operation Directive.** This directive takes effect when you use any Version Manager command.

Syntax `[No]Semaphore [local] [network] = [semaphore_type]`

Use the keyword local for archives stored on a local drive and the keyword *network* for archives stored on a network.

There are two possible values for *semaphore_type*: *netware* and *file*:

- **NetWare** semaphores are created using Novell NetWare API calls. NetWare automatically deletes them if a command terminates abnormally. This type of semaphore is only created for files that are being modified. You can use NetWare semaphores on local files (see note below).
- **File** semaphores are files that Version Manager creates to signal other processes that a file is in use. Version Manager creates file semaphores for files that are being modified, as well as for files that are being read. Always use file semaphores if your environment is composed of heterogeneous network systems

Default Semaphore is the default setting, which means that the program creates file semaphores for local and network files.

Notes When using NetWare semaphores on local files a USERID is not associated. Thus, user A can lockout user B from accessing a file on user B's local drive.

Examples

- The following example creates NetWare semaphores for network files and turns off semaphores for local files:

```
Semaphore network = netware
NoSemaphore local
```

- The following example creates NetWare semaphores for local and network files:
  `Semaphore = netware`
- The following example creates file semaphores for network files:
  `Semaphore network = file`
- The following statement creates file semaphores for both local and network files:
  `Semaphore local network = file`
- The following statement turns off semaphores:
  `NoSemaphore`

Special Considerations

- All users of an archive must use the same type of semaphore.
- **For UNIX users:** Only file semaphores are available under UNIX. Use NoSemaphore to turn off file semaphores.

Related Topics

For information about...	See...
Changing the time between file access attempts	"SemaphoreRetry directive" on page 172
Specifying the directory in which semaphores are created	"SemaphoreRetry directive" on page 172
Changing how many times Version Manager attempts to access a file	"SemaphoreRetry directive" on page 172
Changing the semaphore file extension	"SemSuffix directive" on page 173
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

SemaphoreDelay directive

Specify delay between archive access attempts

Use the SemaphoreDelay directive in conjunction with the Semaphore directive to set the delay between attempts to gain exclusive access to archives, journal files, and the access control database.

Directive type

Operation Directive. This directive takes effect when you use any Version Manager command.

Syntax

`SemaphoreDelay = tenth_of_secs`

Use the *number* parameter to specify the length of the delay in tenths of a second.

Default

The default is SemaphoreDelay = 10 (one second).

Example The following example sets the delay to four seconds:

```
SemaphoreDelay = 40
```

Related Topics

For information about...	See...
Enabling semaphores	"SemaphoreRetry directive" on page 172
Specifying the directory in which semaphores are created	"SemaphoreRetry directive" on page 172
Changing how many times Version Manager attempts to access a file	"SemaphoreRetry directive" on page 172
Changing the semaphore file extension	"SemSuffix directive" on page 173
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

SemaphoreDir directive

Specify directory for semaphore files Use the SemaphoreDir directive to specify the directory in which Version Manager creates semaphore files for archives, journal files, and the access control database.

Directive type **Operation Directive.** This directive takes effect when you use any Version Manager command.

Syntax `SemaphoreDir path`

The *path* parameter must specify a directory that already exists.

Default By default, Version Manager creates file semaphores in the archive directory.

Example The following master configuration file excerpt specifies a semaphore directory. The SemaphoreDir directive is then disallowed, so that users cannot define a different semaphore directory:

DOS

```
SemaphoreDir \serena\sem
Disallow SemaphoreDir
```

UNIX

```
SemaphoreDir /usr/serena/sem
Disallow SemaphoreDir
```

Special Consideration All users of an archive should use the same directory for semaphores. If two users specify different semaphore directories using this directive, they will lose all semaphore protection. For this reason, you should use the SemaphoreDir command in the master configuration file, and then disallow it, as in the example above.

.__$ A semaphore is created with the same file extension mask as that of the archives with a $ replacing the v.

Related Topics

For information about...	See...
Enabling semaphores	"SemaphoreRetry directive" on page 172
Changing the time between file access attempts	"SemaphoreRetry directive" on page 172
Changing how many times Version Manager attempts to access a file	"SemaphoreRetry directive" on page 172
Changing the semaphore file extension	"SemSuffix directive" on page 173
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

SemaphoreRetry directive

Specify number of archive access attempts

Use the SemaphoreRetry directive in conjunction with the Semaphore directive to specify how many attempts Version Manager makes to gain exclusive access to an archive or journal file in network environments.

Directive type

Operation Directive. This directive takes effect when you use any Version Manager command.

Syntax

```
SemaphoreRetry = number
```

The *number* parameter specifies the number of attempts Version Manager makes to access the file before returning an error message.

Default

The default is three attempts.

SemaphoreRetry=3

Example

The following example directs Version Manager to try to gain exclusive access to a file five times before abandoning the effort:

```
SemaphoreRetry = 5
```

Related Topics

For information about...	See...
Enabling semaphores	"SemaphoreRetry directive" on page 172
Changing the time between file access attempts	"SemaphoreRetry directive" on page 172
Specifying the directory in which semaphores are created	"SemaphoreRetry directive" on page 172
Changing the semaphore file extension	"SemSuffix directive" on page 173
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

SemSuffix directive

Specify a semaphore extension

Use the SemSuffix directive to determine the extension of semaphore files.

Version Manager derives the name of the semaphore from the archive name. By default, an archive named TEST.C_V has a semaphore named TEST.C_$. If this is not appropriate for your situation, use the SemSuffix directive to specify a different suffix template.

See "Suffix Translation" on page 123 for more information.

Directive type

Operation Directive. This directive takes effect when you use any Version Manager command.

Syntax

SemSuffix = suffix_template

Use the *suffix_template* parameter to specify the formula by which Version Manager derives semaphore extensions from archive extensions.

Default

The default suffix template for the semaphore name is ??$___.
SemSuffix=??$___

Special Consideration

The SemSuffix directive has an additional syntax for appending a string to file names to form semaphore names:
SemSuffix +*string*

For example, if the directive were SemSuffix +,s and the file name were *foobar.c*, the resulting semaphore name would be *foobar.c,s*.

Related Topics

For information about...	See...
Enabling semaphores	"SemaphoreRetry directive" on page 172
Changing the time between file access attempts	"SemaphoreRetry directive" on page 172
Specifying the directory in which semaphores are created	"SemaphoreRetry directive" on page 172
Changing the number of times Version Manager retries file access	"SemaphoreRetry directive" on page 172
Using suffix templates	"Suffix Translation" on page 123
Using directives	"Configuration Files" on page 111

SignOn directive

Suppress the sign-on message

Use the SignOn directive to control whether Version Manager commands display copyright and version information.

Directive type

Operation Directive. This directive takes effect when you use any Version Manager primary or secondary command.

Syntax `[No]SignOn`

Default The default is SignOn.

Examples When you use the VCS command with the SignOn directive in effect, you see this message:

```
PVCS Version Manager (vcs) v8.6.1.0 (Build 162) for Windows/x86_64
Copyright (C) 1985-2018 Serena Software, Inc., a Micro Focus company.
    All rights reserved.
```

If the NoSignOn directive is in effect, you see this message:

```
Modified archive "archive".
```

Special Consideration You can also use the Diagnostic directive to toggle the sign-on message for all commands, or the -#4 command-line option to toggle it from the command line.

Related Topics

For information about...	See...
Another way to toggle the sign-on message	"Disallow directive" on page 143
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

Translate directive

Translate the end-of-line character Use the Translate directive to control whether Version Manager performs end-of-line (EOL) translation or character set translation.

On UNIX systems, the Translate directive causes Version Manager to translate the end-of-line sequence from line feed to carriage return plus linefeed when you check in a revision, and to make the same translation in reverse when you check a revision out. Doing so lets you use the same archives on UNIX and DOS systems. If you don't need to move archives between systems, don't use the Translate directive.

Directive type **Archive Creation Directive.** This directive only affects archives when they are created. If you change this directive in your configuration file, it has no effect on existing archives. After an archive has been created, use VCS +PT to enable translation and VCS -PT to disable translation.

Syntax `[No]Translate [.ext...]`

Use the optional *.ext* parameter to apply this directive only to files with a certain extension. Enter a period (.) to apply the directive to files with no extension. Enter an asterisk (.*) to apply the directive to all files, regardless of their extension. Enter a question mark (.?) to apply the directive to files with extensions for which there is no defined or default value.

Multiple commands are collective as long as the extensions are different. For example, the following command sets NoTranslate for .EXE and .DLL files:

```
NoTranslate .EXE
NoTranslate .DLL
```

Default — The default is NoTranslate, except for the following file types: .c, .h, .pas, .mak, .for, .bas (all platforms); .asm, .bat (Windows platforms). See "Default Directive Settings" on page 108 for more information.

Special Consideration — **For Windows users:** The Translate directive is currently ignored under Win32. However, the VLOG archive summary reflects the status of this directive for the sake of consistency.

Related Topics

For information about...	See...
Changing translation for existing files	"VCS command" on page 56
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

VCSDir directive

Specify archive directories — Use the VCSDir directive to specify the directories where Version Manager looks for archives. You can also use this directive to create reference directories that correspond to archive directories.

Version Manager searches for archives by looking from left to right in the directories listed in the VCSDir directive. If the archive is not found in a directory in the list, the current directory is searched.

If any directory specified in the VCSDir directive does not exist, Version Manager terminates the command.

When creating archives, if you do not specify where to create an archive, Version Manager creates it in the first directory specified in the VCSDir directive.

Directive type — **Operation Directive.** This directive takes effect when you use a Version Manager command that looks for an archive.

Syntax — `VCSDir = [;]dir_name[(ref_dir)]...`

The directories in the *dir_name* parameter can be absolute or relative path names, as shown below: To specify a directory with a space in the name, enclose the directory name in quotes. For example, to specify the d:\test space directory, enter the following command:

```
VCSDIR = "d:\test space"
```

Default — The default value for VCSDir is the current directory.

```
VCSDir=$(CWD)
```

DOS
```
VCSDir ..\vcs; ..\proj\vcs
```

UNIX
```
VCSDir ../vcs; ../proj/vcs
```

You can specify multiple VCSDir directives for a cumulative effect, where each new VCSDir directive adds more directories to the list. To do so, precede the first directory in subsequent VCSDir directives with a semicolon (;).

The optional *ref_dir* parameter is the name of the reference directory for each *dir_name* directory. Append an empty set of parentheses to a directory name if you do not want a reference directory for that archive directory.

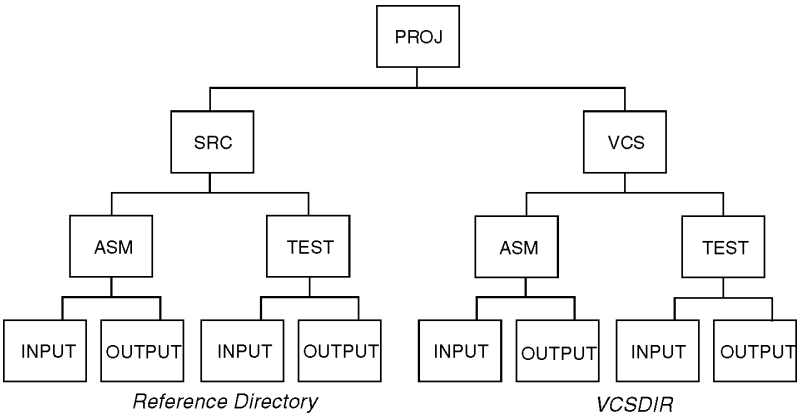
Parallel Reference Directories

Matching archive and reference directories

If your archive directories have a hierarchical structure, you can use the VCSDir directive to maintain a parallel hierarchy for reference directories.

The following figure shows an example of parallel archive and reference hierarchies:

Figure 2-1. Parallel Archive and Reference Hierarchies



Examples

- The following syntax tells Version Manager to use all subdirectories of J:\PROJ\VCS for Windows, */proj/vcs* for UNIX as possible archive directories:

DOS
```
VCSDir j:\proj\vcs\*
```

UNIX
```
VCSDir /proj/vcs/*
```

Use the ReferenceDir directive with the VCSDir syntax above to obtain parallel archive and source directory structures:

DOS
```
ReferenceDir j:\src\*
```

UNIX
```
ReferenceDir /src/*
```

When you use the PUT command, Version Manager creates reference directories for each archive directory in the VCSDir hierarchy beginning at the point symbolized by the asterisk (*).

Special Considerations

- The reference directory must be different from the directory containing the workfile.
- If you use an asterisk (*) to specify a parallel directory hierarchy, you must do so both in the specification for the archive directories and in the specification for the reference directories.

- Deleting directories from your archive hierarchy does not automatically delete the corresponding reference directory.
- If you do not specify a reference directory for an archive directory, Version Manager uses the value specified by the ReferenceDir directive, if defined.
- If you do not specify a directory parameter for the ReferenceDir directive, but only specify a keyword option then that option will be applied to all reference directories specified with the VCSDir directive. For example:

  ```
  ReferenceDir WriteProtect
  VCSDIR=Dir1(Ref1); Dir2(Ref2);....
  ```

  applies *WriteProtect* to *Ref1* and to *Ref2*.
- Using the VCSDir directive controls how Version Manager interprets file specifications. For example, references to *.C are treated differently than references to *.C_V. The command GET *.C means "For all the .C files in the local directory, check out the workfile from the corresponding archive in the VCSDir directory." Conversely, GET *.C_V means "Check out the workfile for all .C_V archives in VCSDir directories."

Related Topics

For information about...	See...
An alternate method of defining reference directories	"Renumber directive" on page 167
Specifying directory parameters	"ReferenceDir directive" on page 166
Changing the file search order	"How Version Manager Searches for Archives" on page 15 "IgnorePath directive" on page 153
Specifying files	"Specifying File Names" on page 10 "Specifying Names Using Wildcards" on page 13
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

VCSEdit directive

Specify the external editor

Use the VCSEdit directive to specify an external text editor for entering workfile or change descriptions. If you don't use this directive, the program provides a simple internal line editor.

Before loading an external editor, Version Manager copies two prompt lines to a temporary file. When the editor runs, these prompt lines remind you what description you are entering. As long as you don't change these lines, Version Manager deletes them for you automatically.

If you modify these lines, the program assumes they are part of your entry. Version Manager deletes the temporary file after use.

Directive type

Operation Directive. This directive takes effect when you use the PUT or VCS command.

Syntax `VCSEdit = editor_command [editor_option...] $temp_file$`

Use the *editor_command* parameter to specify the command that invokes the editor from the command line. Use the *editor_option* parameter to specify any additional command-line options for this editor. Use the required *temp_file* parameter to specify the name and location of the temporary file used for the prompt lines. The *temp_file* is used for the editor's input and output.

Default Notepad for Windows and vi for UNIX.

Example The following example loads Edit instead of the internal editor and creates the temporary file VCSEDIT.TMP.

For UNIX, it loads vi instead of the internal editor and creates the temporary file vcsedit.tmp:

DOS
```
VCSEdit = edit $vcsedit.tmp$
```

UNIX
```
VCSEdit = vi $vcsedit.tmp$
```

Related Topics

For information about...	See...
Commands that use the editor	"PUT command" on page 38 "VCS command" on page 56
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

VCSID directive

Identify the current user Use the VCSID directive to specify the name Version Manager uses to identify revisions and locks in archives. You use the LogIn directive or the VCONFIG -I command to specify this directive as the source Version Manager uses to obtain user identification.

Directive type **Operation Directive.** This directive takes effect when you use any Version Manager primary or secondary command.

Syntax `VCSID = user_id`

Default None

Example
```
VCSID = lamontc
```

Related Topics

For information about...	See...
Specifying the source of user IDs	"LogIn directive" on page 156 "VCONFIG command" on page 52
Making user IDs case-insensitive	"Case directive" on page 128
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

WorkDir directive

Specify the temporary file directory

Use the WorkDir directive to specify the directory where Version Manager creates temporary files. Version Manager usually keeps its internal data structures resident in memory. However, if memory becomes scarce, Version Manager temporarily transfers portions of its internal data structures to disk.

This directive is used for temporary files which are generated during delta processing and other archive file manipulations. This directory is not where the temporary backup archive files are saved. Refer to the ArchiveWork directive for details.

Directive type

Operation Directive. This directive takes effect when you use any Version Manager command.

Syntax

```
WorkDir = path_name
```

Default

The default WorkDir is the current working directory.

Special Considerations

- You can store temporary files on a RAM disk to increase Version Manager performance when running under DOS.
- If you are using a Version Manager File Server, do not set WorkDir to a directory that is mapped to the file server.

Related Topics

For information about...	See...
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

WriteProtect directive

Write-protect archives

Use the WriteProtect directive to write protect archives. Doing so protects archives from inadvertent deletion or modification.

Version Manager commands automatically remove write-protection before modifying the archive and replace it afterwards.

Directive type

Archive Creation Directive. This directive only affects archives when they are created. If you change this directive in your configuration file, it has no effect on existing archives. After an archive has been created, use VCS +PW to enable write protection and VCS -PW to disable write protection.

Syntax

```
[No]WriteProtect
```

Default

The default is WriteProtect.

Related Topics

For information about...	See...
Changing the write-protection attribute for existing archives	"VCS command" on page 56
Using directives	"Configuration Files" on page 111
Other directives	"Using Directives" on page 105

Chapter 3

Security and Privileges

Using Privileges

Control access to Version Manager features

Privileges are keywords that you use in an access control database to control which features a user or group of users has permission to use. You can use the privileges that are built into Version Manager or define custom privilege sets to suit your needs.

Privileges give very specific control over the access rights of each user. If you allow users only those access privileges needed to accomplish their assigned tasks, development becomes much easier to manage.

This section contains the following topics about working with privileges:

- Privilege Types
- Base Privileges, Composite Privileges, and Privilege Sets
- Privilege Access Limits
- Access Control Databases.

Privilege Types

Types of privileges

Version Manager supports the following types of privileges:

- Base privileges are built into Version Manager and represent the access rights to specific features.
- Composite privileges are also built into Version Manager and are composed of two or more related base privileges.
- Privilege sets are divided into two categories, default and custom. Default privilege sets are built into Version Manager and cannot be modified. Custom privilege sets are privileges that you define in the access control database. They can be composed of base privileges, composite privileges, and other privilege sets and can be optionally restricted by promotion group. Several custom privilege sets are provided as samples in the default access control database of the desktop client.

Restrict privileges with the No prefix

You can deny the access right to most base or composite privileges by using the prefix No. For example, the base privilege DeleteRevNonTip allows a user to delete non-tip revisions. The base privilege NoDeleteRevNonTip denies that privilege.

Base Privileges

Provide access to basic features

Each base privilege controls one Version Manager action, although that action might be used by several related Version Manager commands. Also, a Version Manager action might require more than one base privilege. The following table lists base privileges.

Use this base privilege...	To let users do this...
AddGroup	Assign promotion groups to revisions.
AddVersion	Assign version labels. **NOTE** Combine with DeleteVersion to rename version labels.

Use this base privilege...	To let users do this...
BreakLock	Unlock revisions.
ChangeAccessList	Identify users who can access archives.
ChangeCommentDelimiter	Change comment prefixes.
ChangeOwner	Change archive owners.
ChangeProtection	Change archive attributes.
ChangeWorkfileName	Change workfile names.
DeleteGroup	Disassociate promotion groups from revisions.
DeleteRevNonTip	Delete non-tip revisions.
DeleteRevTip	Delete tip revisions.
DeleteVersion	Delete version labels. **NOTE** Combine with AddVersion to rename version labels.
GetNonTip	Check out revisions other than the tip.
GetTip	Check out tip revisions.
InitArchive	Create archives.
JournalReport	View an audit trail of modifications to archives.
LockNonTip	Lock non-tip revisions.
LockTip	Lock tip revisions.
ModifyChangeDescription	Edit change descriptions.
ModifyGroup	Rename/move promotion groups.
ModifyVersion	Re-assign version labels to a different revision, including changing whether the label floats with the tip revision. **NOTE** This privilege does **not** affect your ability to add, delete, or rename version labels. **TIP** Renaming version labels requires **both** the AddVersion and DeleteVersion privileges.
ModifyWorkfileDescription	Change workfile descriptions.
Promote	Promote revisions.
PutBranch	Store branch revisions.
PutTrunk	Store trunk revisions.
StartBranch	Start branches.
Unlock	Remove locks.
ViewAccessDB	View the access control database.
ViewArchiveHeader	View archive header information.
ViewArchiveRev	View revision histories.

Composite Privileges

Privileges built from other privileges

Composite privileges consist of two or more base privileges. The rights of a composite privilege are the sum of the access rights of its component base privileges. The following table lists the composite privileges and their component base privileges.

Use this composite privilege...	To let users do this...
DeleteRev	DeleteRevNonTip DeleteRevTip
Get	GetNonTip GetTip
Lock	LockNonTip LockTip
ModifyDescription	ModifyChangeDescription ModifyWorkfileDescription
Put	PutBranch PutTrunk
ViewArchive	ViewArchiveHeader ViewArchiveRev

Privilege Sets

Collections of privileges

There are two types of privilege sets, default and custom. Version Manager provides several default privilege sets, which are defined in below. The default privilege sets, SuperUser and Unlimited, cannot be modified.

Custom privilege sets are named collections of base, composite, and other custom privilege sets. Version Manager allows you to define a custom privilege set so that you can create privilege sets that are suited to your working environment.

Custom Privilege Sets

Define custom privileges

You define *custom privilege sets* to combine a group of privileges into a single task-oriented privilege set. By defining custom privilege sets, you can give individual users or members of a group access to only the privileges they need. You can use existing privilege sets (default and custom) as part of the definition of new custom privilege sets.

Default Privilege Sets

Default privilege sets provided

The following table lists the default and custom privilege sets that are available with Version Manager. Remember that only the SuperUser and Unlimited privilege sets are provided with the command-line interface. The remaining privilege sets are provided as samples that can be viewed using the desktop client. The table also lists the base, composite, and other privilege sets that comprise the default privilege sets.

Privilege Set	Base Privileges
SuperUser (built-in)	All base privileges on all archives
Unlimited (built-in)	All base privileges on all accessible archives

Privilege Set	Base Privileges
Developer (desktop client only)	Project Lead privilege set minus Create Project, Configure Project, Copy Project, Delete Project, Add or Remove Versioned Files, Modify Project, and Configure Security
Project Lead (desktop client only)	Lock Tip, Lock Non-Tip, Unlock, Check Out Tip, Check Out Non-tip, Check In Tip, Check In Branch, Create Branch, Modify Archive Access List, Modify Archive Owner, Modify Archive Attributes, Modify Comment Delimiter, Modify Versioned File Name, Modify Workfile Description, Modify Change Description, Add Version Label, Delete Version Label, Modify Version Label, Create Archive, Delete Tip, Delete Non-tip, View Archive Header, View Archive Revisions, Promote to Next Promotion Group, Add Promotion Group, Modify Promotion Group, Delete Promotion Group, Create Project, Configure Project, Copy Project, Delete Project, Add or Remove Versioned Files, Modify Project, Configure Security, and Show Journal
Documentation (desktop client only)	Quality Assurance privilege set plus Lock Tip, Lock Non-tip, Unlock, Check Out Tip, Check Out Non-tip, Check in Tip, Check In Branch, and Create Archive
Quality Assurance (desktop client only)	Support privilege set plus Add Version Label, Delete Version label, Modify Version Label, Promote to Next Promotion Group, Add Promotion Group, Modify Promotion Group, and Delete Promotion Group
Support (desktop client only)	View Archive Header, View Archive Revisions, and Show Journal

SuperUser grants all rights

The SuperUser privilege set grants unlimited access to any archive, whether or not the user is on the access list. SuperUser cannot be assigned to a group; it requires a user definition. SuperUser is the only privilege set that cannot be restricted by using the NO prefix.

Unlimited grants base and composite privileges

The Unlimited privilege set grants all Version Manager base and composite privileges. By default, all users have the Unlimited privilege set. If a user has been assigned the Unlimited privilege set in the access control database, access rights to archives are limited only by access lists.

Privilege Access Limits

Rules for determining privileges

Version Manager privileges are governed by the following rules:

- By default, users have the Unlimited privilege set. If a user has the Unlimited privilege assigned in the access control database, access rights to archives are limited only by access lists.
- If a user has the SuperUser privilege set assigned in the access control database, access rights are unlimited (can access any archive), regardless of access list membership.
- The SuperUser privilege set cannot be assigned to a group—only to individual users.

- Privileges assigned to a group can only limit the privileges of the users of the group, not increase the privileges of the users of the group.

For more information on how privileges affect access control, see Chapter 2, *Using Security* in the <Emphasis>PVCS Version Manager Administrator's Guide.

Example

The following example includes privilege, group, and user definitions:

```
privilege view: get,viewarchiveheader,viewarchiverev
privilege audit: viewarchiveheader,viewarchiverev,getnontip
user charlesd (view)
user janea (audit)
group library (view): charlesd,janea
```

The group library has the View privilege. Even though user janea is a member of this group, she still has the Audit privilege. Therefore, she does not have the GetTip privilege (a component of the Get privilege in View), even though her group has that privilege because group privileges cannot expand an individual user's assigned privileges.

NOTE

- The names of base privileges and composite privileges are not case-sensitive.
- The names of custom privilege sets are case-sensitive.
- The SuperUser privilege set is a "blank check" for all Version Manager operations. You should assign this privilege set only to a limited number of administrative users.

Related Topics

For information about each privilege, see the topic for that privilege.

Access Control Databases

Control archive access

The *access control database* is an encrypted file that defines the users who are authorized to perform actions on archives and defines the actions that users can perform. The access control database contains user IDs, privileges, and access list groups. This database also contains user ID/password pairs used by the VLOGIN source.

NOTE The VLOGIN login source applies only to the desktop client. See the VCONFIG command and LogIn directive for more information on login sources.

You can create an access control database from the command line by:

- Creating an ordinary text file that contains privileges, groups, and users,
- Creating the access control database,
- Specifying the access control database location.

For more information on how to create an access control database, see the *Administrator's Guide*.

NOTE To associate access lists with archives or to use the VLOGIN source to provide password protection, you must have an access control database.

Access control text file format

The format of an access control text file is similar to a Version Manager configuration file. Certain guidelines apply when creating access control databases from text files. See the *Administrator's Guide* for more information.

Identify the access control database name and location

Once you have created the access control database, you specify the name and location where you want Version Manager to store it. You can specify the name and location in two ways:

- Embed the name and location of the access control database in the VCONFIG file using the VCONFIG -A command. For Windows, the file name is VMWFVC.DLL; for UNIX, vmufvc.a.
- Edit the master configuration file and identify the access control database name using the AccessDB directive. For added security, use the Disallow directive to disallow the AccessDB directive after you use it, so others cannot change the access control database name in their own configuration files and possibly circumvent security.

For more information on specifying an access control database name and location, see the *Administrator's Guide*.

NOTES

- A default name and location does not exist for the access control database. You must specify the name of the file using the VCONFIG -A command or the AccessDB directive.
- If users have access to the AccessDB directive, they can define a different location for the access control database than the one defined with VCONFIG -A. For maximum security, disallow the use of AccessDB.
- By default, Version Manager user IDs are case sensitive. If you define user names in the access control database in lowercase letters, and your network returns user IDs in uppercase letters, Version Manager won't recognize the user IDs as those defined in the access control database. To correct this problem, set the directive NoCase=VCSID in the master configuration file.

Related Topics

For information about...	See...
Encrypting the access control database text file	"MAKEDB command" on page 35
Commands related to access control	"READDB command" on page 44 "VCONFIG command" on page 52
Assigning an access list to a new archive	"AccessList directive" on page 118
Using base and composite privileges	"Using Privileges" on page 183
Specifying the name and location of the access control database	"AccessDB directive" on page 117
Creating a master configuration file	"Master Configuration Files" on page 110
Changing access list entries for new and existing archives	"PUT command" on page 38 "VCS command" on page 56
Limiting the use of a directive	"Disallow directive" on page 143

The remainder of this chapter contains a detailed description of each privilege in alphabetical order.

AddGroup privilege

Assign promotion groups to revisions

Use the AddGroup privilege to let users assign promotion groups to revisions.

Privilege type

Base Privilege. The AddGroup privilege lets you use the `vcs -g` command.

Syntax

```
[No]AddGroup
```

Related Topics

For information about...	See...
Commands affected by this privilege	"VCS command" on page 56
Other privileges	"Using Privileges" on page 183

AddVersion privilege

Assign version labels to revisions

Use the AddVersion privilege to let users attach version labels to revisions.

Privilege type

Base Privilege. The AddVersion privilege allows you to use the `vcs -v` and `put -v` commands.

Syntax

```
[No]AddVersion
```

TIP Renaming version labels requires **both** the AddVersion and DeleteVersion privileges.

Related Topics

For information about...	See...
Commands affected by this privilege	"PUT command" on page 38 "VCS command" on page 56
Related privileges	"DeleteVersion privilege" on page 194 "ModifyVersion privilege" on page 201 "Put privilege" on page 209 "PutBranch privilege" on page 209 "PutTrunk privilege" on page 210
Other privileges	"Using Privileges" on page 183

BreakLock privilege

Lets you unlock revisions

Use the BreakLock privilege to let a user unlock a revision locked by someone else.

Privilege type **Base Privilege**. The BreakLock privilege allows you to use the `vcs -u` command.

Syntax `[No]BreakLock`

Special Consideration Don't confuse BreakLock with the Unlock privilege, which only lets you remove locks on revisions locked by your user ID.

Related Topics

For information about...	See...
Commands affected by this privilege	"VCS command" on page 56
Related privileges	"Unlock privilege" on page 213
Other privileges	"Using Privileges" on page 183

ChangeAccessList privilege

Lets you change access lists Use the ChangeAccessList privilege to let users change archive access lists.

Privilege type **Base Privilege**. The ChangeAccessList privilege allows you to use the `vcs -a` and `put -a` commands.

Syntax `[No]ChangeAccessList`

Related Topics

For information about...	See...
Commands affected by this privilege	"PUT command" on page 38 "VCS command" on page 56
Related privileges	"InitArchive privilege" on page 197 "Put privilege" on page 209
Other privileges	"Using Privileges" on page 183

ChangeCommentDelimiter privilege

Lets you change the comment prefix Use the ChangeCommentDelimiter privilege to let users change the comment prefix used in keyword expansion.

Privilege type **Base Privilege**. The ChangeCommentDelimiter privilege lets you use the `vcs -ec` command.

Syntax `[No]ChangeCommentDelimiter`

Related Topics

For information about...	See...
Commands affected by this privilege	"VCS command" on page 56
Specifying comment characters for Log expansion	"CommentPrefix directive" on page 130
Using keywords	*Administrator's Guide*
Related privileges	"ChangeProtection privilege" on page 191
Other privileges	"Using Privileges" on page 183

ChangeOwner privilege

Lets you change archive owners

Use the ChangeOwner privilege to let users change archive owners.

Privilege type

Base Privilege. The ChangeOwner privilege allows you to use the `vcs -o` command.

Syntax `[No]ChangeOwner`

Related Topics

For information about...	See...
Commands affected by this privilege	"VCS command" on page 56
Other privileges	"Using Privileges" on page 183

ChangeProtection privilege

Lets you change archive protection

Use the ChangeProtection privilege to let users modify the following characteristics of archives:

- Whether revision locking is enabled
- Whether more than one revision can be locked at a time
- Whether keywords are expanded
- Whether translation is performed on UNIX systems
- Whether archives are write-protected
- Whether information in archives is compressed
- Whether delta records are generated

Privilege type

Base Privilege. The ChangeProtection privilege allows you to use the VCS PC, PD, PE, PG, PK, PL, PT, and PW commands.

Syntax `[No]ChangeProtection`

Related Topics

For information about...	See...
Commands affected by this privilege	"VCS command" on page 56
Other privileges	"Using Privileges" on page 183

ChangeWorkfileName privilege

Lets you change workfile names

Use the ChangeWorkfileName privilege to let users change the name of the workfile stored in an archive.

Privilege type

Base Privilege. The ChangeWorkfileName privilege allows you to use the `vcs -w` command.

Syntax `[No]ChangeWorkfileName`

Related Topics

For information about...	See...
Commands affected by this privilege	"VCS command" on page 56
Other privileges	"Using Privileges" on page 183

DeleteGroup privilege

Remove promotion groups from revisions

Use the DeleteGroup privilege to let users delete promotion groups from revisions. This privilege and the ModifyGroup privilege must be in effect to rename or move a promotion group.

Privilege type

Base Privilege. The DeleteGroup privilege lets you use the `vcs -gpromotion_group:delete` command.

Syntax `[No]DeleteGroup`

Related Topics

For information about...	See...
Commands affected by this privilege	"PUT command" on page 38 "VCS command" on page 56
Related privileges	"AddGroup privilege" on page 189
Other privileges	"Using Privileges" on page 183

DeleteRev privilege

Lets you delete revisions

Use the DeleteRev privilege to let users delete revisions from archives. However, you cannot delete a revision that has a promotion group associated with it.

Privilege type

Composite Privilege. The DeleteRev privilege is composed of the DeleteRevTip and DeleteRevNonTip base privileges. It allows you to use the VDEL command.

Syntax

```
[No]DeleteRev
```

Related Topics

For information about...	See...
Commands affected by this privilege	"VDEL command" on page 64
Related privileges	"DeleteRevNonTip privilege" on page 193 "DeleteRevTip privilege" on page 193
Other privileges	"Using Privileges" on page 183

DeleteRevNonTip privilege

Lets you delete non-tip revisions

Use the DeleteRevNonTip privilege to let users delete non-tip revisions from archives.

Privilege type

Base Privilege. The DeleteRevNonTip privilege lets you use the `vdel -r` command.

Syntax

```
[No]DeleteRevNonTip
```

Related Topics

For information about...	See...
Commands affected by this privilege	"VDEL command" on page 64
Related privileges	"DeleteWork directive" on page 137 "DeleteRevTip privilege" on page 193
Other privileges	"Using Privileges" on page 183

DeleteRevTip privilege

Lets you delete tip revisions

Use the DeleteRevTip privilege to let users delete tip revisions.

Privilege type

Base Privilege. The DeleteRevTip privilege allows you to use the VDEL -R command.

Syntax `[No]DeleteRevTip`

Related Topics

For information about...	See...
Commands affected by this privilege	"VDEL command" on page 64
Related privileges	"DeleteWork directive" on page 137 "DeleteRevNonTip privilege" on page 193
Other privileges	"Using Privileges" on page 183

DeleteVersion privilege

Lets you delete version labels

Use the DeleteVersion privilege to let users delete version labels from an archive.

Privilege type

Base Privilege. The DeleteVersion privilege allows you to use the `vcs -v<version_label>:delete` command.

Syntax `[No]DeleteVersion`

TIP Renaming version labels requires **both** the AddVersion and DeleteVersion privileges.

Related Topics

For information about...	See...
Commands affected by this privilege	"VCS command" on page 56
Related privileges	"AddVersion privilege" on page 189 "ModifyVersion privilege" on page 201
Other privileges	"Using Privileges" on page 183

Developer privilege (desktop client only)

Lets you perform developer tasks

Use the Developer privilege set to grant users privileges to tasks most commonly performed by developers. This default custom privilege set is provided as a sample, and it can be accessed using the access control database of the desktop client.

Privilege type

Default Privilege Set. The Developer privilege set is composed of the Project Lead privilege set minus the Create Project, Configure Project, Copy Project, Delete Project, Add or Remove Versioned Files, Modify Project, and Configure Security privileges.

Related Topics

For information about...	See...
Default and custom privilege sets	"Privilege Sets" on page 185
Other privilege sets	"Documentation privilege (desktop client only)" on page 195 "Project Lead privilege (desktop client only)" on page 208 "Quality Assurance privilege (desktop client only)" on page 210 "SuperUser privilege" on page 211 "Support privilege (desktop client only)" on page 212 "Unlimited privilege" on page 213
Other privileges	"Using Privileges" on page 183

Documentation privilege (desktop client only)

Lets you perform documentation tasks

Use the Documentation privilege set to grant users privileges to tasks most commonly performed by documentation group members. This custom privilege set is provided as a sample, and it can be accessed using the access control database of the desktop client.

Privilege type

Default Privilege Set. The Documentation privilege set is composed of Quality Assurance privilege set plus Lock Tip, Lock Non-tip, Unlock, Check Out Tip, Check Out Non-tip, Check in Tip, Check In Branch, and Create Archive privileges.

Related Topics

For information about...	See...
Default and custom privilege sets	"Privilege Sets" on page 185
Other privilege sets	"Developer privilege (desktop client only)" on page 194 "Project Lead privilege (desktop client only)" on page 208 "Quality Assurance privilege (desktop client only)" on page 210 "SuperUser privilege" on page 211 "Support privilege (desktop client only)" on page 212 "Unlimited privilege" on page 213
Other privileges	"Using Privileges" on page 183

Get privilege

Lets you retrieve a revision

Use the Get privilege to let users check out revisions from archives. To lock revisions, users must also have the Lock, LockTip, or LockNonTip privileges.

Privilege type

Composite Privilege. The Get privilege is composed of the GetTip and GetNonTip base privileges. It lets you use the GET and VMRG commands.

Syntax `[No]Get`

Related Topics

For information about...	See...
Related privileges	"GetNonTip privilege" on page 196 "GetTip privilege" on page 196 "LockNonTip privilege" on page 198 "Lock privilege" on page 198 "LockTip privilege" on page 199
Commands affected by this privilege	"GET command" on page 29 "VMRG command" on page 81
Other privileges	"Using Privileges" on page 183

GetNonTip privilege

Lets you retrieve a non-tip revision

Use the GetNonTip privilege to let users check out non-tip revisions.

Privilege type

Base Privilege. The GetNonTip privilege allows you to use the `get -r`, `get -v`, `vmrg -r`, and `vmrg -v` commands.

Syntax `[No]GetNonTip`

Related Topics

For information about...	See...
Related privileges	"Get privilege" on page 195 "GetTip privilege" on page 196 "LockNonTip privilege" on page 198 "Lock privilege" on page 198 "LockTip privilege" on page 199
Commands affected by this privilege	"GET command" on page 29 "VMRG command" on page 81
Other privileges	"Using Privileges" on page 183

GetTip privilege

Lets you retrieve a tip revision

Use the GetTip privilege to let users check out tip revisions.

Privilege type

Base Privilege. The GetTip privilege lets you use the `get -r`, `get -v`, `put -u`, `put -l`, `vmrg -r`, and `vmrg -v` commands.

Syntax `[No]GetTip`

Related Topics

For information about...	See...
Related privileges	"Get privilege" on page 195 "GetNonTip privilege" on page 196 "LockNonTip privilege" on page 198 "Lock privilege" on page 198 "LockTip privilege" on page 199
Commands affected by this privilege	"GET command" on page 29 "VMRG command" on page 81
Other privileges	"Using Privileges" on page 183

InitArchive privilege

Lets you create an archive

Use the InitArchive privilege to let users create archives.

Privilege type

Base Privilege. The InitArchive privilege allows you to use the `put` and `vcs -i` commands to create archives.

Syntax `[No]InitArchive`

Special Considerations

- This privilege overrides the AutoCreate directive.
- The InitArchive privilege was formerly called the InitLogfile privilege. Version Manager still recognizes it by that name.

Related Topics

For information about...	See...
Commands affected by this privilege	"PUT command" on page 38 "VCS command" on page 56
Directives affected by this privilege	"AutoCreate directive" on page 126
Other privileges	"Using Privileges" on page 183

JournalReport privilege

Lets you view an archive's audit trail

Use the JournalReport privilege to let users view an audit trail of all modifications made to an archive using Version Manager commands.

Privilege type

Base Privilege. The JournalReport privilege allows you to use the VJOURNAL command.

Syntax `[No]JournalReport`

Special Consideration

The JournalReport privilege was formerly called the ActionsJournalReport privilege. Version Manager still recognizes it by that name.

Related Topics

For information about...	See...
Commands affected by this privilege	"VJOURNAL command" on page 73
Directives affected by this privilege	
Other privileges	"Using Privileges" on page 183

Lock privilege

Lock revisions

Use the Lock privilege to let users lock revisions.

Privilege type

Composite Privilege. The Lock privilege is composed of the LockTip and LockNonTip base privileges. It lets you use the `get -l`, `put -l`, and `vcs -l` commands.

Syntax

```
[No]Lock
```

Related Topics

For information about...	See...
Commands affected by this privilege	"GET command" on page 29 "PUT command" on page 38 "VCS command" on page 56
Related privileges	"Get privilege" on page 195 "LockNonTip privilege" on page 198 "LockTip privilege" on page 199 "Put privilege" on page 209

LockNonTip privilege

Lets you lock non-tip revisions

Use the LockNonTip privilege to let users lock a revision other than the tip on any branch or on the trunk.

Privilege type

Base Privilege. The LockNonTip privilege lets you use the `get -l`, `put -l`, and `vcs -l` commands.

Syntax `[No]LockNonTip`

Related Topics

For information about...	See...
Commands affected by this privilege	"GET command" on page 29 "PUT command" on page 38 "VCS command" on page 56
Related privileges	"Get privilege" on page 195 "GetNonTip privilege" on page 196 "GetTip privilege" on page 196 "Lock privilege" on page 198 "LockTip privilege" on page 199 "PutBranch privilege" on page 209 "Put privilege" on page 209 "StartBranch privilege" on page 211

LockTip privilege

Lets you lock tip revisions

Use the LockTip privilege to let users lock trunk and branch tip revisions.

Privilege type

Base Privilege. The LockTip privilege lets you use the `get -l`, `put -l`, and `vcs -l` commands.

Syntax `[No]LockTip`

Related Topics

For information about...	See
Commands affected by this privilege	"GET command" on page 29 "PUT command" on page 38 "VCS command" on page 56
Related privileges	"Get privilege" on page 195 "GetNonTip privilege" on page 196 "GetTip privilege" on page 196 "LockNonTip privilege" on page 198 "Lock privilege" on page 198 "Put privilege" on page 209
Other privileges	"Using Privileges" on page 183

ModifyChangeDescription privilege

Lets you edit change descriptions

Use the ModifyChangeDescription privilege to let users edit change descriptions in archives.

Privilege type **Base Privilege.** This privilege lets you use the `vcs -m` command.

Syntax `[No]ModifyChangeDescription`

Related Topics

For information about...	See...
Commands affected by this privilege	"VCS command" on page 56
Related privileges	"ModifyDescription privilege" on page 200 "ModifyWorkfileDescription privilege" on page 201
Other privileges	"Using Privileges" on page 183

ModifyDescription privilege

Lets you edit workfile or change descriptions Use the ModifyDescription privilege to let users modify workfile or change descriptions.

Privilege type **Composite Privilege.** The ModifyDescription privilege is composed of the ModifyChangeDescription and ModifyWorkfile Description privileges. It lets you use the VCS -T and VCS -M commands.

Syntax `[No]ModifyDescription`

Related Topics

For information about...	See...
Commands affected by this privilege	"VCS command" on page 56
Related privileges	"ModifyChangeDescription privilege" on page 199 "ModifyWorkfileDescription privilege" on page 201
Other privileges	"Using Privileges" on page 183

ModifyGroup privilege

Change promotion groups Use the ModifyGroup privilege to let users change promotion groups associated with a revision.

Privilege type **Base Privilege.** The ModifyGroup privilege allows use of the `vcs -g` or `put -g` command. If a promotion group already exists in an archive and it becomes necessary to rename or move it, this privilege must be allowed.

Syntax `[No]ModifyGroup`

Related Topics

For information about...	See...
Commands affected by this privilege	"PUT command" on page 38 "VCS command" on page 56
Other privileges	"Using Privileges" on page 183
Related privileges	"AddGroup privilege" on page 189 "DeleteMessageFile directive" on page 136

ModifyVersion privilege

Lets you change version labels

Use the ModifyVersion privilege to let users re-assign version labels to a different revision. This includes the ability to change whether a label is Fixed to a specific revision or Floats with the tip revision.

Privilege type **Base Privilege.** The ModifyVersion privilege lets you use the `vcs -v` command.

Syntax `[No]ModifyVersion`

NOTE This privilege does **not** affect your ability to add, delete, or rename version labels.

TIP Renaming version labels requires **both** the AddVersion and DeleteVersion privileges.

Related Topics

For information about...	See...
Commands affected by privilege	"VCS command" on page 56
Related privileges (Renaming version labels.)	"AddVersion privilege" on page 189 "DeleteVersion privilege" on page 194
Other privileges	"Using Privileges" on page 183

ModifyWorkfileDescription privilege

Lets you change workfile descriptions

Use the ModifyWorkfileDescription privilege to let users edit descriptions in new archives and on existing workfiles.

Privilege type **Base Privilege.** The ModifyWorkfileDescription privilege lets you use the `vcs -t` or `vcs -m` command.

Syntax `[No]ModifyWorkfileDescription`

Related Topics

For information about...	See...
Commands affected by this privilege	"VCS command" on page 56
Related privileges	"ModifyChangeDescription privilege" on page 199 "ModifyDescription privilege" on page 200
Other privileges	"Using Privileges" on page 183

NoActionsJournalReport privilege (desktop client only)

Disables the ability to run journal reports

Use the NoActionsJournalReport privilege to disable users from generating journal reports in the 5.3/6.0 desktop client.

Privilege type Project Privilege.

Syntax `NoActionsJournalReport`

Special Consideration The NoActionsJournalReport privilege works only with Version Manager 5.3/6.0 projects. The new desktop client privilege name for this action is Show Journal.

Related Topics

For information about...	See...
Other 5.3/6.0 project privileges	"NoOptionsSecurity privilege (desktop client only)" on page 205, "NoProjectConfigureProject privilege (desktop client only)" on page 206, "NoProjectNewProject privilege (desktop client only)" on page 207, "NoProjectCopyProject privilege (desktop client only)" on page 206, "NoProjectDeleteProject privilege (desktop client only)" on page 207

NoFolderChangeFolder privilege (desktop client only)

Disables users from folder actions

Use the NoFolderChangeFolder privilege to disable users from renaming folders and changing folder attributes in the 5.3/6.0 desktop client.

Privilege type Folder Privilege.

Syntax `NoFolderChangeFolder`

Special Consideration The NoFolderChangeFolder privilege works only with Version Manager 5.3/6.0 projects. This privilege does not appear in the new desktop client.

Related Topics

For information about...	See...
Other 5.3/6.0 folder privileges	"NoFolderChangeFolderMembers privilege (desktop client only)" on page 203, "NoFolderCopyFolderMembers privilege (desktop client only)" on page 203, "NoFolderDeleteFolder privilege (desktop client only)" on page 204, "NoFolderNewFolder privilege (desktop client only)" on page 204, "NoFolderUpdateProjectFolder privilege (desktop client only)" on page 205,

NoFolderChangeFolderMembers privilege (desktop client only)

Disables adding work files and deleting versioned files

Use the NoFolderChangeFolderMembers privilege to disable users from adding workfiles, importing archives, and deleting versioned files from project databases or projects.

Privilege type

Project Privilege.

Syntax

`NoFolderChangeFolderMembers`

Special Consideration

The NoFolderChangeFolderMembers privilege works only with Version Manager 5.3/6.0 projects. The new desktop client privilege name for this action is Add or Remove Versioned Files.

Related Topics

For information about...	See...
Other 5.3/6.0 folder privileges	"NoFolderChangeFolder privilege (desktop client only)" on page 202, "NoFolderCopyFolderMembers privilege (desktop client only)" on page 203, "NoFolderDeleteFolder privilege (desktop client only)" on page 204, "NoFolderNewFolder privilege (desktop client only)" on page 204, "NoFolderUpdateProjectFolder privilege (desktop client only)" on page 205,

NoFolderCopyFolderMembers privilege (desktop client only)

Prevents users from copying folders

Use the NoFolderCopyFolderMembers privilege to disable users from copying folders in the 5.3/6.0 desktop client.

Privilege type

Folder Privilege.

Syntax `NoFolderCopyFolderMembers`

Special Consideration The NoFolderCopyFolderMembers privilege works only with Version Manager 5.3/6.0 projects. This privilege does not appear in the new desktop client.

Related Topics

For information about...	See...
Other 5.3/6.0 folder privileges	"NoFolderChangeFolder privilege (desktop client only)" on page 202, "NoFolderChangeFolderMembers privilege (desktop client only)" on page 203, "NoFolderDeleteFolder privilege (desktop client only)" on page 204, "NoFolderNewFolder privilege (desktop client only)" on page 204, "NoFolderUpdateProjectFolder privilege (desktop client only)" on page 205,

NoFolderDeleteFolder privilege (desktop client only)

Disables users from deleting folders Use the NoFolderDeleteFolder privilege to disable users from deleting folders in the 5.3/6.0 desktop client.

Privilege type Folder Privilege.

Syntax `NoFolderDeleteFolder`

Special Consideration The NoFolderChangeFolder privilege works only with Version Manager 5.3/6.0 projects. This privilege does not appear in the new desktop client.

Related Topics

For information about...	See...
Other 5.3/6.0 folder privileges	"NoFolderChangeFolder privilege (desktop client only)" on page 202, "NoFolderChangeFolderMembers privilege (desktop client only)" on page 203, "NoFolderCopyFolderMembers privilege (desktop client only)" on page 203, "NoFolderNewFolder privilege (desktop client only)" on page 204, "NoFolderUpdateProjectFolder privilege (desktop client only)" on page 205,

NoFolderNewFolder privilege (desktop client only)

Disables users from creating new folders Use the NoFolderNewFolder privilege to disable users from creating folders in the 5.3/6.0 desktop client.

Privilege type Folder Privilege.

Syntax `NoFolderNewFolder`

Special Consideration The NoFolderNewFolder privilege works only with Version Manager 5.3/6.0 projects. This privilege does not appear in the new desktop client.

Related Topics

For information about...	See...
Other 5.3/6.0 folder privileges	"NoFolderChangeFolder privilege (desktop client only)" on page 202, "NoFolderChangeFolderMembers privilege (desktop client only)" on page 203, "NoFolderCopyFolderMembers privilege (desktop client only)" on page 203, "NoFolderDeleteFolder privilege (desktop client only)" on page 204, "NoFolderUpdateProjectFolder privilege (desktop client only)" on page 205,

NoFolderUpdateProjectFolder privilege (desktop client only)

Prevents users from updating project folders Use the NoFolderUpdateProjectFolder privilege to disable users from updating folders in the 5.3/6.0 desktop client.

Privilege type Folder Privilege.

Syntax `NoFolderUpdateProjectFolder`

Special Consideration The NoFolderUpdateProjectFolder privilege works only with Version Manager 5.3/6.0 projects. This privilege does not appear in the new desktop client.

Related Topics

For information about...	See...
Other 5.3/6.0 folder privileges	"NoFolderChangeFolder privilege (desktop client only)" on page 202, "NoFolderChangeFolderMembers privilege (desktop client only)" on page 203, "NoFolderCopyFolderMembers privilege (desktop client only)" on page 203, "NoFolderDeleteFolder privilege (desktop client only)" on page 204, "NoFolderNewFolder privilege (desktop client only)" on page 204,

NoOptionsSecurity privilege (desktop client only)

Disables changes to security settings Use the NoOptionsSecurity privilege to disable users from defining users, groups, privileges, and login sources.

Privilege type Project Privilege.

Syntax `NoOptionsSecurity`

Special Consideration The NoOptionsSecurity privilege works only with Version Manager 5.3/6.0 projects. The new desktop client privilege name for these actions is Configure Security.

Related Topics

For information about...	See...
Other 5.3/6.0 project privileges	"NoActionsJournalReport privilege (desktop client only)" on page 202, "NoProjectConfigureProject privilege (desktop client only)" on page 206, "NoProjectCopyProject privilege (desktop client only)" on page 206, "NoProjectDeleteProject privilege (desktop client only)" on page 207, "NoProjectNewProject privilege (desktop client only)" on page 207

NoProjectConfigureProject privilege (desktop client only)

Disables project configuration and workspace setting modification Use the NoProjectConfigureProject privilege to disable users from configuring projects and from modifying the base, branch, and default version settings for workspaces.

Privilege type Project Privilege.

Syntax `NoProjectConfigureProject`

Special Consideration The NoProjectConfigureProject privilege works only with Version Manager 5.3/6.0 projects. The new desktop client privilege name for these actions is Configure Project.

Related Topics

For information about...	See...
Other 5.3/6.0 project privileges	"NoActionsJournalReport privilege (desktop client only)" on page 202, "NoOptionsSecurity privilege (desktop client only)" on page 205, "NoProjectNewProject privilege (desktop client only)" on page 207, "NoProjectCopyProject privilege (desktop client only)" on page 206, "NoProjectDeleteProject privilege (desktop client only)" on page 207

NoProjectCopyProject privilege (desktop client only)

Disables project copy Use the NoProjectCopyProject privilege to disable users from copying projects.

Privilege type: Project Privilege.

Syntax: `NoProjectCopyProject`

Special Consideration: The NoProjectCopyProject privilege works only with Version Manager 5.3/6.0 projects. The new desktop client privilege name for this action is Copy Project.

Related Topics

For information about...	See...
Other 5.3/6.0 project privileges	"NoActionsJournalReport privilege (desktop client only)" on page 202, "NoOptionsSecurity privilege (desktop client only)" on page 205, "NoProjectConfigureProject privilege (desktop client only)" on page 206, "NoProjectDeleteProject privilege (desktop client only)" on page 207, "NoProjectNewProject privilege (desktop client only)" on page 207

NoProjectDeleteProject privilege (desktop client only)

Disables project deletion: Use the NoProjectDeleteProject privilege to disable users from deleting projects.

Privilege type: Project Privilege.

Syntax: `NoProjectDeleteProject`

Special Consideration: The NoProjectDeleteProject privilege works only with Version Manager 5.3/6.0 projects. The new desktop client privilege name for this action is Delete Project.

Related Topics

For information about...	See...
Other 5.3/6.0 project privileges	"NoActionsJournalReport privilege (desktop client only)" on page 202, "NoOptionsSecurity privilege (desktop client only)" on page 205, "NoProjectConfigureProject privilege (desktop client only)" on page 206, "NoProjectCopyProject privilege (desktop client only)" on page 206, "NoProjectNewProject privilege (desktop client only)" on page 207

NoProjectNewProject privilege (desktop client only)

Disables project creation: Use the NoProjectNewProject privilege to disable users from creating projects.

Privilege type: Project Privilege.

Syntax `NoProjectNewProject`

Special Consideration The NoProjectNewProject privilege works only with Version Manager 5.3/6.0 projects. The new desktop client privilege name for this action is Create Project.

Related Topics

For information about...	See...
Other 5.3/6.0 project privileges	"NoActionsJournalReport privilege (desktop client only)" on page 202, "NoOptionsSecurity privilege (desktop client only)" on page 205, "NoProjectConfigureProject privilege (desktop client only)" on page 206, "NoProjectCopyProject privilege (desktop client only)" on page 206, "NoProjectDeleteProject privilege (desktop client only)" on page 207

Project Lead privilege (desktop client only)

Lets you perform project lead tasks Use the Project Lead privilege set to grant users privileges to tasks most commonly performed by project leads. This custom privilege set is provided as a sample, and it can be accessed using the access control database of the desktop client.

Privilege type **Default Privilege Set**. The Project Lead privilege set is composed of Lock Tip, Lock Non-tip, Unlock, Check Out Tip, Check Out Non-tip, Check In Tip, Check In Branch, Create Branch, Modify Archive Access List, Modify Archive Owner, Modify Archive Attributes, Modify Comment Delimiter, Modify Versioned File Name, Modify Workfile Description, Modify Change Description, Add Version Label, Delete Version Label, Modify Version Label, Create Archive, Delete Tip, Delete Non-tip, View Archive Header, View Archive Revisions, Promote to Next Promotion Group, Add Promotion Group, Modify Promotion Group, Delete Promotion Group privileges, Create Project, Modify Project, Delete Project, Copy Project, Add or Remove Versioned Files, Configure Project, Configure Security, and Show Journal.

Related Topics

For information about...	See...
Default and custom privilege sets	"Privilege Sets" on page 185
Other privilege sets	"Developer privilege (desktop client only)" on page 194 "Documentation privilege (desktop client only)" on page 195 "Quality Assurance privilege (desktop client only)" on page 210 "SuperUser privilege" on page 211 "Support privilege (desktop client only)" on page 212 "Unlimited privilege" on page 213
Other privileges	"Using Privileges" on page 183

Promote privilege

Lets you promote revisions

Use the Promote privilege to let users promote revisions from one promotion group to the next higher promotion group.

Privilege type

Base Privilege.

Syntax

```
[No]Promote
```

Related Topics

For information about...	See...
Commands affected by this privilege	"VPROMOTE command" on page 84
Other privileges	"Using Privileges" on page 183

Put privilege

Lets you store revisions

Use the Put privilege to let users check in branch or trunk revisions.

Privilege type

Composite Privilege. The Put privilege is composed of the PutBranch and the PutTrunk privileges. It lets you use the PUT command.

Syntax

```
[No]Put
```

Special Consideration

You can only create an archive if you have the InitArchive privilege. You can only start a branch if you have the StartBranch privilege.

Related Topics

For information about...	See...
Commands affected by this privilege	"PUT command" on page 38
Related privileges	"Get privilege" on page 195 "LogIn directive" on page 156 "Lock privilege" on page 198 "PutBranch privilege" on page 209 "PutTrunk privilege" on page 210 "StartBranch privilege" on page 211
Other privileges	"Using Privileges" on page 183

PutBranch privilege

Lets you store branch revisions

Use the PutBranch privilege to let users check in branch revisions.

Privilege type

Base Privilege. The PutBranch privilege lets you use the PUT command when the revision is on a branch.

Syntax `[No]PutBranch`

Special Consideration To start a branch, you must also have the StartBranch privilege.

Related Topics

For information about...	See...
Commands affected by this privilege	"PUT command" on page 38
Related privileges	"Get privilege" on page 195 "Lock privilege" on page 198 "Put privilege" on page 209 "PutTrunk privilege" on page 210 "StartBranch privilege" on page 211
Other privileges	"Using Privileges" on page 183

PutTrunk privilege

Lets you store trunk revisions Use the PutTrunk privilege to let users check in trunk revisions.

Privilege type **Base Privilege**. The PutTrunk privilege lets you use the PUT command when the revision is on the trunk.

Syntax `[No]PutTrunk`

Related Topics

For information about...	See...
Commands affected by this privilege	"PUT command" on page 38
Related privileges	"Get privilege" on page 195 "Lock privilege" on page 198 "Put privilege" on page 209 "PutBranch privilege" on page 209 "StartBranch privilege" on page 211
Other privileges	"Using Privileges" on page 183

Quality Assurance privilege (desktop client only)

Lets you perform quality assurance tasks Use the Quality Assurance privilege set to grant users privileges to tasks most commonly performed by quality assurance group members. This custom privilege set is provided as a sample, and it can be accessed using the access control database of the desktop client.

Privilege type **Default Privilege Set**. The Quality Assurance privilege set is composed of the Support privilege set plus Add Version Label, Delete Version Label, Modify Version Label, Promote to Next Promotion Group, Add Promotion Group, Modify Promotion Group, and Delete Promotion Group privileges.

Related Topics

For information about...	See...
Default and custom privilege sets	"Privilege Sets" on page 185
Other privilege sets	"Developer privilege (desktop client only)" on page 194 "Documentation privilege (desktop client only)" on page 195 "Project Lead privilege (desktop client only)" on page 208 "SuperUser privilege" on page 211 "Support privilege (desktop client only)" on page 212 "Unlimited privilege" on page 213
Other privileges	"Using Privileges" on page 183

StartBranch privilege

Lets you start a branch

Use the StartBranch privilege to allow users to begin a branch from any revision.

Privilege type

Base Privilege. The StartBranch privilege allows you to use the `put -fb` command.

Syntax

```
[No]StartBranch
```

Related Topics

For information about...	See...
Commands affected by this privilege	"PUT command" on page 38
Related privileges	"Put privilege" on page 209 "PutBranch privilege" on page 209
Other privileges	"Using Privileges" on page 183

SuperUser privilege

Lets you use any command on any archive

Use the SuperUser privilege set to grant users unlimited access to all archives. This privilege set is a default privilege set and may not be modified.

Privilege type

Default Privilege Set. The SuperUser privilege set is composed of all base privileges plus the following actions: create, rename, and delete public workspaces; copy and rename project databases; and not check in workfiles when adding workfiles. This privilege set is not restricted by access lists.

Special Considerations

- The SuperUser privilege set is a "blank check" for all Version Manager operations. You should only assign this privilege set to a limited number of administrative users.
- Only assign this privilege to users, not groups or you could get unexpected permissions.
- You cannot use the No modifier with this privilege.

Related Topics

For information about...	See...
Another privilege that grants all privileges	"Unlimited privilege" on page 213
Privilege categories	"Privilege Sets" on page 185
Other privilege sets	"Developer privilege (desktop client only)" on page 194 "Documentation privilege (desktop client only)" on page 195 "Project Lead privilege (desktop client only)" on page 208 "Quality Assurance privilege (desktop client only)" on page 210 "Support privilege (desktop client only)" on page 212 "Unlimited privilege" on page 213
Other privileges	"Using Privileges" on page 183

Support privilege (desktop client only)

Lets you perform quality assurance tasks

Use the Support privilege set to grant users privileges to tasks most commonly performed by support group members. This custom privilege set is provided as a sample, and it can be accessed using the access control database of the desktop client.

Privilege type

Default Privilege Set. The Support privilege set is composed of the View Archive Header, View Archive Revisions, and Show Journal.

Related Topics

For information about...	See...
Default and custom privilege sets	"Privilege Sets" on page 185
Other privilege sets	"Developer privilege (desktop client only)" on page 194 "Documentation privilege (desktop client only)" on page 195 "Project Lead privilege (desktop client only)" on page 208 "Quality Assurance privilege (desktop client only)" on page 210 "SuperUser privilege" on page 211 "Unlimited privilege" on page 213
Other privileges	"Using Privileges" on page 183

Unlimited privilege

Lets you use any command

Use the Unlimited privilege set to give users all Version Manager privileges on any archive for which one of the following is true:

- Their user IDs appear in the archive's access list.
- The archive access list is empty.

This privilege set is a default composite privilege set and may not be modified.

Privilege type

Default Privilege Set. The Unlimited privilege set is composed of all Version Manager base privileges plus the following actions: create, rename, and delete public workspaces and not check in workfiles when adding workfiles.

Syntax

```
[No]Unlimited
```

Related Topics

For information about...	See...
Other privileges	"Using Privileges" on page 183
Privilege categories	"Privilege Sets" on page 185
Other privilege sets	"Developer privilege (desktop client only)" on page 194 "Documentation privilege (desktop client only)" on page 195 "Project Lead privilege (desktop client only)" on page 208 "Quality Assurance privilege (desktop client only)" on page 210 "SuperUser privilege" on page 211 "Support privilege (desktop client only)" on page 212
Privilege sets	"Privilege Sets" on page 185

Unlock privilege

Lets you remove locks

Use the Unlock privilege to allow users to unlock any revision that was locked by their user IDs.

Privilege type

Base Privilege. The Unlock privilege allows you to use the `vcs -u` command.

Syntax

```
[No]Unlock
```

Special Consideration

Do not confuse this privilege with the BreakLock privilege, which allows you to remove any lock regardless of the user ID associated with that lock.

Related Topics

For information about...	See...
Commands affected by this privilege	"VCS command" on page 56
Related privileges	"BreakLock privilege" on page 189
Other privileges	"Using Privileges" on page 183

ViewAccessDB privilege

Lets you view the access control database

Use the ViewAccessDB privilege to allow users to view information contained in the access control database.

Privilege type

Base Privilege. The ViewAccessDB privilege allows you to use the READDB command.

Syntax

```
[No]ViewAccessDB
```

Special Consideration

The ViewAccessDB privilege was formerly called the ActionsSecurity privilege. Version Manager still recognizes it by that name.

Related Topics

For information about...	See...
Commands affected by this privilege	"READDB command" on page 44
Other privileges	"Using Privileges" on page 183

ViewArchive privilege

Lets you view archive information

Use the ViewArchive privilege to allow users to view information about archives.

Privilege type

Composite Privilege. The ViewArchive privilege is composed of the ViewArchiveHeader and ViewArchiveRev base privileges. It allows you to use the VLOG command.

Syntax

```
[No]ViewArchive
```

Special Consideration

In earlier versions of the program, the ViewArchive privilege was named ViewLogfile. Version Manager still recognizes that name for it.

Related Topics

For information about...	See...
Commands affected by this privilege	"VLOG command" on page 76
Related privileges	"ViewArchiveHeader privilege" on page 214 "ViewArchiveRev privilege" on page 215
Other privileges	"Using Privileges" on page 183

ViewArchiveHeader privilege

Lets you view archive header information

Use the ViewArchiveHeader privilege to allow users to view archive header information, but not the revision information.

Privilege type **Base Privilege**. The ViewArchiveHeader privilege allows you to use the `vlog -b` command.

Syntax `[No]ViewArchiveHeader`

Special Consideration In earlier versions of the program, the ViewArchiveHeader privilege was named ViewLogfileHeader. Version Manager still recognizes that name.

Related Topics

For information about...	See...
Commands affected by this privilege	"VLOG command" on page 76
Related privileges	"ViewArchive privilege" on page 214 "ViewArchiveRev privilege" on page 215
Other privileges	"Using Privileges" on page 183

ViewArchiveRev privilege

Lets you view archive revision history Use the ViewArchiveRev privilege to allow users to view the revision history in archives.

Privilege type **Base Privilege**. The ViewArchiveRev privilege allows you to use the following commands:

vlog -c
vlog -d
vlog -i
vlog -l
vlog -o
vlog -q
vlog -r
vlog -br

Syntax `[No]ViewArchiveRev`

Special Consideration In earlier versions of the program, the ViewArchiveRev privilege was named ViewLogfileRev. Version Manager still recognizes that name.

Related Topics

For information about...	See...
Commands affected by this privilege	"VLOG command" on page 76
Related privileges	"ViewArchiveHeader privilege" on page 214 "ViewArchive privilege" on page 214
Other privileges	"Using Privileges" on page 183

Appendix A: Naming Conventions and Restrictions

General Naming Conventions and Restrictions

You can use most alpha, numeric, and special characters when creating or renaming Version Manager entities and paths. However, your operating system also determines the conventions that apply to file and directory names.

Prohibited Characters for Files and Directories

The following characters are prohibited by Version Manager, and most operating systems, when naming files or directories:

- Angle brackets (>) and (<)
- Asterisk (*)
- Colon (:)
- Pipe (|)
- Question mark (?)
- Quotation mark (")
- Slashes, forward (/) and backward (\)
- Space ( ) as the first or last character
- Tab

IMPORTANT! On Windows systems, files and directories (and thus Version Manager entities and paths) cannot end with a period (.).

Naming Considerations for Cross-Platform Environments

When working in a cross-platform environment, be aware of any incompatibilities between the systems and limit your usage to that which they have in common.

IMPORTANT! In a cross-platform environment, you cannot place files into the same directory if they differ only be case. Such usage is possible only in UNIX-only environments.

Specific Naming Conventions and Restrictions

The following table lists naming conventions and restrictions that apply to specific Version Manager entities and paths.

	Restrictions	
Item Type	**Characters**	**Length**
Archives	As listed in "Prohibited Characters for Files and Directories" on page 218, plus cannot use: Ampersand: & Brackets: [ ] Parenthesis: ( ) Plus sign: + Semicolon: ;	254 (full path including the file name) **NOTE** Only the first 10 characters of the archive suffix are significant in distinguishing identically named files in the same project.
Files and paths (unless otherwise noted)	As listed in "Prohibited Characters for Files and Directories" on page 218.	254 (full path including the file name)
pvcs_bindir	As listed in "Prohibited Characters for Files and Directories" on page 218.	254 (full path including the file name) **NOTE** On UNIX, the name of the `vconfig` file and the separator character also count against the total length.
Project databases	Cannot begin or end with a tab or space character. Any character can be used within the name.	
Projects	As listed in "Prohibited Characters for Files and Directories" on page 218, plus cannot be: ■ The two-character name of: .. ■ The one-character name of: . ■ The one-character name of: @	

	Restrictions	
Item Type	**Characters**	**Length**
Promotion groups	Ampersand: & Brackets: [ ] Comma: , Equal sign: = Parenthesis: ( ) Plus sign: + Question mark: ? Semicolon: ; Slash: /	
User, group, and privilege names	Asterisk: * Colon: : Backward slash: \ Single quote: ' Quotation mark: " Parenthesis: ( )	30
Version labels	Ampersand: & Asterisk: * Brackets: [ ] Colon: : Equal sign: = Minus sign: - Parenthesis: ( ) Plus sign: + Question mark: ? Quotation mark: " Semicolon: ; Slash: / **NOTE** The backslash (\) serves as an escape character. To create or delete a label that includes a backslash, the backslash must be preceded by another backslash (\\Label would result in \Label; \\\\Label would result in \\Label).	254

Appendix B: Error Messages

Version Manager error messages

This appendix lists the error messages generated by Version Manager. Codes generic to all commands are on page 221, specific error messages are listed alphabetically starting on page 221, internal error messages are on page 231. Messages that begin with a specific filename are listed under "F"; messages that begin with a specific directive name are listed under "D."

To get additional information about an error message, first consult the alphabetic list of messages. If you don't find the message, check the list of internal error messages. If you still do not find the exact error message, look again for a similarly worded message.

Version Manager will issue many of its messages in one of the formats below, where *command* is the command you entered, and *message* is the error message listed in this appendix.

```
command: warning, message
command: error, message
```

Exit Codes Generic to All Commands

Exit code 128 (Windows)

An exit code of 128 occurs when a CLI command cannot find the DTK DLL, probably because the PATH environment variable does not include *installDir*`\vm\common\bin\win32`.

If the error occurs while you are using CLI commands outside of Configuration Builder, you see a pop-up error message referring to `vmwfdtk.dll`. If you are running a Configuration Builder command and you specify `(ignore)` so that the script does not terminate on errors, you receive no message, and the command returns error code 128.

Error Messages

Archive "*filename*" contains no revisions

You issued a command that required an archive containing one or more revisions, but the specified archive contains no revision. Make sure that you specified the correct archive.

For more information about...	See...
Specifying archives	"Specifying File Names" on page 10

Archive "*filename*" is in use

A semaphore exists for this archive, indicating that it is currently in use. The file may actually be in use, or there may be a semaphore file left over from a network crash during a Version Manager operation.

If the archive is actually busy, wait until the other operation is complete and try again. If you are sure that no one else is accessing the archive, remove the semaphore file.

For information about...	See...
Specifying a semaphore location	"SemaphoreRetry directive" on page 172

Archive "*filename*" cannot be modified by "*user_name*"

This user ID is not on the access list for the archive. Make sure that you're using the correct user ID, or modify the access list.

Badly formed date argument "*date/time*"

You used a date/time specification that was syntactically incorrect.

For information about...	See...
Specifying dates and times	"Specifying Dates and Times" on page 23

Badly formed revision "*revision*"

You specified a revision number that was inconsistent or syntactically incorrect.

For information about...	See...
Specifying revisions	"Specifying Revisions in Commands" on page 18
How revisions are numbered	"Specifying a Revision Number" on page 39

"-c" option must occur first

You used the -C command-line option to specify an alternate configuration file, but you didn't use it as the first command-line option, as required.

For information about...	See...
Using the -C option	"Configuration Files" on page 111
Commands that use the -C option	All commands except FILT, IDENT, PRINTENV, and RSE

Cannot change attributes of file "*filename*" to *hex_number*

Version Manager tried unsuccessfully to modify the attributes of a workfile or archive. You may have insufficient network access rights to the file. Another cause can be a file system with an incompatible device driver.

For information about...	See...
Changing network access rights	Your network documentation

Cannot create workfile|archive "*filename*"
Cannot delete file "*filename*"
Cannot open workfile|archive "*filename*" for *operation*

The operating system returned an error code during the operation, probably because you do not have sufficient access rights to the file.

For information about...	See...
Changing network access rights	Your network documentation

Cannot execute edit command -"*editor_command*"

Version Manager encountered a problem when attempting to load an alternative text editor. Make sure that the command to invoke the editor is correct. This message may also indicate insufficient memory.

For information about...	See...
Solving editor memory problems	"VCSEdit directive" on page 177

Cannot find argument file "*filename*"

You used the `@filename` command-line option to specify a list file, but Version Manager can't find the file. Verify the location of the list file.

For information about...	See...
Using list files	"List and Group Files" on page 21
Commands that use the @ option	All commands except FILT, PRINTENV, and RSE

Cannot find file "*filename*"

Version Manager cannot find the specified file. Verify the location of the file, and make sure that you have sufficient access rights to it.

For information about...	See...
How Version Manager finds files	"How Version Manager Searches for Archives" on page 15

Cannot find archive for "*workfile*"

Version Manager can't find the archive that corresponds to the workfile you specified. Verify the location of the archive, and make sure that Version Manager is looking in the right place for it.

For information about...	See...
Specifying archives	"Specifying File Names" on page 10
How Version Manager finds files	"How Version Manager Searches for Archives" on page 15
Directives that affect file searching	"VCSDir directive" on page 175 "IgnorePath directive" on page 153
How Version Manager determines archive names	"ArchiveSuffix directive" on page 122

Cannot find revision *revision* in archive "*filename*"

The archive doesn't contain the specified revision. Make sure that the revision and archive specifications are correct. If necessary, use the VLOG command to determine which revisions the archive contains.

For information about...	See...
Specifying archives and workfiles	"Specifying File Names" on page 10
Specifying revisions	"Specifying Revisions in Commands" on page 18

Cannot locate archive "*filename*"

Version Manager can't find the archive you specified in any of the VCSDir directories, the current directory, or the specified directory.

Make sure the archive specification is correct. Also check the VCSDir and ArchiveSuffix directives that apply to this archive.

For information about...	See...
Specifying archives	"Specifying File Names" on page 10
How Version Manager finds files	"How Version Manager Searches for Archives" on page 15
Directives that affect file searching	"VCSDir directive" on page 175 "IgnorePath directive" on page 153
How Version Manager determines archive names	"ArchiveSuffix directive" on page 122

Cannot locate archive for workfile "*filename*"

Version Manager can't find the archive you specified. Make sure the archive specification is correct. Also check the VCSDir and ArchiveSuffix directives that apply to this archive.

For information about...	See...
Specifying workfiles	"Specifying File Names" on page 10
How Version Manager finds files	"How Version Manager Searches for Archives" on page 15
Directives that affect file searching	"VCSDir directive" on page 175 "IgnorePath directive" on page 153
How Version Manager determines archive names	"ArchiveSuffix directive" on page 122

Cannot locate workfile "*workfile_name*"

Version Manager can't find the archive you specified in the current or specified directory.

For information about...	See...
Specifying workfiles	"Specifying File Names" on page 10
How Version Manager finds files	"Specifying Revisions in Commands" on page 18

Cannot open file "*filename*" for reading
Cannot open file "*filename*" for writing

Version Manager can't open the file, probably because it doesn't exist. Check the name and location of the file.

For information about...	See...
Specifying filenames	"Specifying File Names" on page 10
How Version Manager finds files	"How Version Manager Searches for Archives" on page 15

Cannot open temporary edit file "*filename*"

Before invoking the editor specified by the VCSEdit directive, Version Manager attempted to write a prompt to a temporary file but was denied access to the file. You may have insufficient access rights to the temporary file directory. You can specify a different location for the temporary file as a parameter to the VCSEdit directive.

For information about...	See...
Invoking an external editor	"VCSEdit directive" on page 177
Changing network access rights	Your network documentation

Cannot access journal file "*filename*"

Version Manager attempted unsuccessfully to write to this journal file. The journal file may have been write-protected, or you may have insufficient network access rights to it.

For information about...	See...
When Version Manager writes to journal files	"LogIn directive" on page 156
Changing network access rights	Your network documentation

Cannot create journal file semaphore "*semaphore_file*"

In a network environment where the Semaphore directive is in effect, Version Manager tried repeatedly but unsuccessfully to access the journal file.

This situation arises when extremely heavy network traffic or some other condition causes Version Manager to keep the journal file open for longer than 20 to 40 seconds. Wait for a while, then try the command again.

This message may also mean that a semaphore exists even when no one is accessing the journal file. This may happen if the network server crashed while Version Manager was writing to the journal file. If this is the case, delete the semaphore and continue.

For information about...	See...
When Version Manager writes to journal files	"LogIn directive" on page 156

Configuration file nesting exceeded *number*

The nesting of configuration files exceeds the maximum allowable level of 20. This is usually caused when two configuration files each include the other, forming a recursive loop.

To discover configuration file loops, use the -#1 command-line option or the Diagnostic = 1 directive to display the configuration file names as they are processed.

For information about...	See...
Nesting configuration files	"Include directive" on page 154
Displaying diagnostic information	"Disallow directive" on page 143
Commands that use the -# directive	All commands except FILT, IDENT, PRINTENV, and RSE

Directive directive not supported in *version* Version Manager - ignored

This directive is not valid for the current version of Version Manager. Remove it from the configuration file.

For information about...	See...
Directives supported in this version	"Using Directives" on page 105

Error exit from editor -"*editor_command*"

The editor command specified for the VCSEdit directive returned a non-zero exit code on termination. Use the editor in a way that avoids the non-zero exit code, or use the internal editor.

For information about...	See...
Invoking an external editor	"VCSEdit directive" on page 177

Error on closing archive|workfile "*filename*"
Error on reading archive|workfile "*filename*"
Error on writing archive|workfile "*filename*" (disk full?)

The operating system returned an error code while accessing this file. This may be caused by disk or drive failure, or by insufficient disk space. Make sure that the disk and drive are operating correctly and that there is sufficient free disk space for the operation.

For information about...	See...
Diagnosing disk or drive problems	Your operating system or computer manual

File must not be a directory (*filename*)

You issued a Version Manager command with a directory name where a filename should have been.

For information about...	See...
General information about using commands	"Using Commands" on page 8
Specifying archive and workfile names	"Specifying File Names" on page 10

"*filename*" is not a directory

In the Version Manager configuration file, you used this filename as a parameter to a directive that requires a directory name. For information on individual directives, see the topic for each directive.

For information about...	See...
Editing configuration files	"Using Commands" on page 8

"*filename*" is not a Version Manager archive

You specified a file that is not an archive. Check the name and location of the archive.

For information about...	See...
Specifying archive names	"Specifying File Names" on page 10

Group *addition* not allowed by *user*.
Group *delete* not allowed by *user*.
Group *modify* not allowed by *user*.

You attempted to add, delete, or modify a group without the privileges to do so.

For information about...	See...
Assigning privileges	"Using Privileges" on page 183
Group privileges	"AddGroup privilege" on page 189 "DeleteMessageFile directive" on page 136 "ModifyGroup privilege" on page 200

Illegal path separator character - ignored

The configuration file contains a PathSeparator directive that uses an invalid character as the path component separator.

For information about...	See...
Valid path separator characters	"PathSeparator directive" on page 163
Editing configuration files	"Configuration Files" on page 111

Invalid VCSEDIT directive

The configuration file contains a VCSEdit directive that is syntactically incorrect.

For information about...	See...
Correct syntax for the VCSEdit directive	"VCSEdit directive" on page 177
Editing configuration files	"Configuration Files" on page 111

Insufficient memory to continue processing

There is not enough memory available to run the command. Remove memory-resident programs to make more memory available, or add more memory to your system.

For information about...	See...
Removing memory-resident programs	The documentation for those programs

Invalid argument "*argument*" ignored

You issued a command with an unknown or unsupported argument. For information on using individual commands, see the topic for each command.

For information about...	See...
General information about using commands	"Using Commands" on page 8

Invalid date component in "*date/time*"

The date/time specification contained an inconsistent element, such as a month greater than 12 or a day too large for the month.

For information about...	See...
Specifying dates and times	"Specifying Dates and Times" on page 23

Locking "*archive*" rev *revision* would cause a branch when checked in

The BranchWarn directive is in effect, and Version Manager has detected an imminent branch. Determine whether you want the branch and take appropriate action.

For information about...	See...
Disabling the BranchWarn directive	"BranchWarn directive" on page 127

No files match "*wildcard_specification*"

You used a wildcard file specification for which there were no matching files.

For information about...	See...
Using wildcard file specifications	"Specifying Names Using Wildcards" on page 13
Specifying archives and workfiles	"Specifying File Names" on page 10
How Version Manager finds files	"How Version Manager Searches for Archives" on page 15

No files specified

You didn't specify any files for the command.

For information about...	See...
Specifying archives and workfiles	"Specifying File Names" on page 10

No files processed

Either you didn't specify any files for the command, or you used a wildcard specification for which there were no matching files.

For information about...	See...
Using wildcard file specifications	"Specifying Names Using Wildcards" on page 13
Specifying archives and workfiles	"Specifying File Names" on page 10
How Version Manager finds files	"How Version Manager Searches for Archives" on page 15

Range end not allowed when range start is a branch

You used the VDEL command to specify an incorrect range of revisions.

For information about...	See...
Specifying revision ranges	"Specifying Revisions in Commands" on page 18
Using the VDEL command	"VDEL command" on page 64

Revision missing on -r option

You used the -R command-line option without specifying a revision number.

For information about...	See...
Specifying revisions	"Specifying Revisions in Commands" on page 18
Commands that use the -R option	"GET command" on page 29 "PUT command" on page 38 "VCS command" on page 56 "VDEL command" on page 64 "VDIFF command" on page 67 "VLOG command" on page 76 "VMRG command" on page 81

Template missing on -S option

You used the -S command-line option without specifying an alternate archive suffix template.

For information about...	See...
Specifying a suffix template	"Suffix Translation" on page 123
Commands that use the -S option	"GET command" on page 29 "PUT command" on page 38 "VCS command" on page 56 "VDEL command" on page 64 "VDIFF command" on page 67 "VLOG command" on page 76 "VMRG command" on page 81

Unknown directive in file "*filename*" - ignored

The configuration file contains an invalid or misspelled directive. Make sure that the file is really a configuration file, and that it contains valid directives.

For information about...	See...
Valid directives	"Using Directives" on page 105

Unrecognized option "*parameter*"

You used a command-line option that is not valid for the command. For information about each command, see the topic for that command.

For information about...	See...
Using commands	"Using Commands" on page 8

Unknown modifier for EXPANDKEYWORDS - ignored
Unknown modifier for NODELETEWORK - ignored
Unknown modifier for SCREEN - ignored
Unknown modifier in file *"filename"* - ignored

This directive was followed by an unrecognized keyword.

For information about...	See...
Directive keywords	"DeleteWork directive" on page 137 "ExpandKeywords Touch directive" on page 150

Version "*version_label*" not found in archive "*filename*"

You used a version label that does not exist in this archive. Make sure that you specified the right archive and version label.

For information about...	See...
Specifying archive names	"Specifying File Names" on page 10
Specifying version labels	"Specifying Version Labels in Commands" on page 17

Version Manager archive "*filename*" is corrupted

This file appears to be an archive but it contains an inconsistency. Make sure that the archive specification is correct. If necessary, restore the archive from a backup copy.

For information about...	See...
Specifying archive names	"Specifying File Names" on page 10

Virtual file swap (write) error (disk full?)

The operating system returned an error code when Version Manager tried to write a temporary file to the directory specified by the WorkDir directive.

Make sure that the WorkDir directive specifies a directory to which you have access rights, and that there is enough room on the drive. The temporary file may be as large as the sum of the archive size plus three times the workfile size.

For information about...	See...
Specifying a temporary file location	"WorkDir directive" on page 179

Workfile and archive cannot be the same ("*filename*")

You entered a command where the specified or inferred archive and workfile names were the same.

Make sure that you specified the right archive and workfile names. Also, check that the ArchiveSuffix directive is appropriate for your file naming conventions.

For information about...	See...
Specifying workfile and archive names	"Specifying File Names" on page 10
Specifying archive extensions	"ArchiveSuffix directive" on page 122
How Version Manager infers archive extensions	"Suffix Translation" on page 123

Internal Error Messages

Code-related error messages

If one of the following error messages appears, please contact support. See "Contacting Technical Support" on page 10.

```
Bad case value in filter() [number]
Cannot seek workfile|archive "filename" to number (number)
Error returned by free()
Internal type number
Unexpected EOF encountered on workfile|archive "filename"
Unexpected error code (number) returned from name
Unknown delta code (number) at offset number in file "filename"
Unknown error (number) in gen_text_rec()
Unknown error code number
Virtual file error (number) on file "filename" (number)
Virtual file system initialization error number
```

Index

Symbols

A

D

E

F

G

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X