MICRO FOCUS | SERENA

PVCS Version Manager

PCLI User's Guide and Reference

Product version: 8.6.1

Publication date: August 2018

Table of Contents

Chapter 1

Using the Project Command-Line Interface

Introduction

The project command-line interface (PCLI) lets you work with project databases and projects using a command-line interface in addition to using the Version Manager desktop client. For example, using the PCLI, you can:

- Create project databases, projects, and workspaces
- Set workfile locations, archive locations, and configuration files
- Get the current workfile locations, archive locations, and configuration file names and locations
- List project databases, versioned files, and the attributes of different entities
- Add workfiles to projects
- Import archives to projects

You can use the commands provided in the PCLI in combination with the commands provided in the existing command-line interface. The PCLI uses a project path to identify files. The existing command-line interface uses both the workfile location and archive location to identify files. The existing command-line interface is documented in the *Command-Line Reference Guide*.

PCLI provides you with the ability to perform multiple actions on projects interactively or using scripts.

This chapter explains the methods you can use to execute PCLI commands, provides a listing and definition of the PCLI commands, and discusses the Version Manager entities (such as projects and versioned files) and command options appropriate for PCLI commands.

The Command Interpreter

The Version Manager PCLI commands are executed by a command interpreter. When using the PCLI commands, you must precede them with `pcli`. For example:
`pcli` **`getconfigfile -h`**

Two methods of executing the PCLI commands are provided:

- Direct
- Batch

Direct Command Execution

Using direct command execution, you can invoke one PCLI command at a time on the command line. In this case, the specified command is executed and then the command interpreter is exited.

The command syntax for direct execution is:

`pcli` *`command_name`* [*`options`*]

where:

command_name is any PCLI command, as listed in Figure 1-1, "PCLI Commands," on page 10.

options are any valid options for the specified command. The option **-h** is valid for all commands. This option displays the online help for the specified command. For example, the following command displays the online help for the GetConfigFile command.

```
pcli getconfigfile -h
```

The valid options for each PCLI command are documented in Chapter 3, "PCLI Command Reference" on page 43, which is an alphabetical reference of commands. A set of common options are available for some of the PCLI commands; see "Common Project Command Options" on page 16.

Batch Command Execution

The batch method of executing commands allows you to execute a PCLI script through the PCLI. The PCLI Run command is provided for this purpose—to read and process a series of commands from a file. A PCLI script is a file that contains a series of PCLI commands.

The syntax to execute a script using the run command is:

```
pcli run command_options -sPCLI_Script [script_arguments]
```

When you use the Run command to execute a PCLI script, the PCLI commands in the script must **not** be preceded by pcli, as they are when you execute them using the direct method of execution. A simple PCLI script might look like this:

```
# Create a project database named productb
createprojectdb -prD:\productb -nproductb -wD:\productb\work
# Create a project named class within productb
createproject -prD:\productb -wD:\productb\work\class class
```

You would execute the script by entering the following:

```
pcli run -sPCLIscript
```

See Chapter 2, "Scripting" on page 33 for detailed information about scripting using the PCLI.

PCLI Commands

PCLI commands are divided into the following functional areas:

- Project
- Scripting

The following table defines each PCLI command and lists its alias, if one is available. Chapter 3, "PCLI Command Reference" on page 43 provides the syntax, valid options, and examples of the use of each of these commands.

Table 1-1. PCLI Commands

Command Command's Alias	Definition
Project Commands	
AddFiles AF	Adds workfiles to projects given the workfile locations.
AddUser AU	Adds a user to a project or project database.
AssignGroup AG	Assigns a specified promotion group to versioned files.
ChangeGroup CG	Specifies a different promotion group for versioned files that already have a promotion group.
CreateProject CP	Creates a project within the specified project database.
CreateProjectDB CPDB	Creates a project database at the location specified.
CreateWorkspace CW	Creates a new private or public workspace.
Delete	Deletes workspaces, projects, folders, and versioned files
DeleteGroup DG	Removes a specified promotion group from versioned files.
DeleteLabel DL	Removes a version label from projects or versioned files.
DeleteUser DU	Deletes a user from the access control database.
ExportPDB EPDB	Exports project database and project information into an INI file format.
Get	Checks out files.
GetArchiveLocation GAL	Lists the archive location for the specified project or versioned file.
GetConfigFile GCF	Lists the configuration files used by the specified project.
GetWorkLocation GWL	Lists the workfile location for the specified project, 5.3/6.0 folder, or versioned file.
ImportArchives IA	Adds versioned files to projects given the archive locations.
ImportPdb IPDB	Creates a project database and projects from the project database information in an INI file.
IsDifferent ID	Tests for differences between workfiles, a workfile and a versioned file, of between revisions of a versioned file.
Label	Assigns a version label to a revision of versioned files.
List	Lists Version Manager entities (such as projects and versioned files) and their attributes.

Command Command's Alias	Definition
ListFloatingLabels LFL	Lists the floating labels and their associated revision numbers for Version Manager entities.
ListProjectDB LPDB	Lists the current project database location or lists all of the active project database locations used by the desktop client.
ListPromotionModel LPM	LIsts the promotion model associated with a Version Manager entity.
LIstRevision LR	Lists the revision associated with the default version or with a specified version label or promotion group.
ListVersionedFiles LVF	Lists all versioned files of a project.
Lock	Locks a revision of the specified versioned files.
Move	Moves versioned items into a destination project database, project, or folder.
PromoteGroup PG	Promotes versioned files to the next promotion group.
Put	Checks in a revision of a versioned file.
RenameLabel RL	Renames a version label in projects or versioned files.
SetArchiveInfo SAI	Sets archive information for versioned files.
SetArchiveLocation SAL	Sets the archive location for the specified project or versioned file.
SetConfigFile SCF	Associates the specified configuration file with a project.
SetWorkLocation SWL	Sets the workfile location for the specified project, 5.3/6.0 folder, or versioned file.
Unlock	Unlocks a revision of a versioned file.
UpdateProjectFolder UPF	Updates the list of project files to reflect changes in the archive directories specified in the VCSDir directive. Applies only to 5.3/6.0 projects.
Vdel	Deletes a revision of a versioned file.
Vlog	Reports archive and revision information of versioned files.
VmCopy VC	Copies a group of items to a project.
WhoAmI WAI	Reports the active user ID used to access specified Version Manager entities.
Scripting Commands	
Run	Executes PCLI scripts, functions, and external programs.
@	Inserts the contents of a file into a script.
Echo	Outputs literal text or the contents of PCLI variables.

Command Command's Alias	Definition
Elif	Used with the If command to specify a boolean condition and the action to be taken if it is met.
Else	Used with the If command to specify the action to be taken if the boolean conditions stated in the If command are not met.
For	Executes a set of commands once for each element in a variable array.
If	Executes one set of commands or another depending on the result of a boolean command.
While	Executes a set of commands as long as the results of a boolean command is true.
Test	Evaluates a boolean expression for the If and While commands.
Calc	Evaluates a numeric expression.
Set	Sets the value of a PCLI variable or variable array.
ArraySize AS	Displays the combined size of the specified variable arrays.
Exit	Stops processing and returns the specified status to the calling program.
Return	Causes the current function or script to return with the specified status.
Break	Exits the current For or While loop.
Continue	Branches to the top of the current For or While loop.
Readline	Enables a PCLI script to prompt the user from the command line. Places a line of user input into a variable.

Version Manager Entities

Version Manager entities are the different items that the PCLI can operate on. The common entities are project database, project, folder, versioned file, and workspace. The description of each PCLI project command in Chapter 3, "PCLI Command Reference" on page 43 defines the valid entities for each of the commands.

Each PCLI project command, except CreateProjectDB, requires that you specify the project database that you want to work with. The two most common ways to specify a project database are to:

- Set the PCLI_PR variable. See "Variables" on page 18.
- Specify the **-pr** option on the command line. See "Common Project Command Options" on page 16.

Also, most PCLI project commands require that you specify a project. The most common ways to specify a project are to:

- Set the PCLI_PP variable. See "Variables" on page 18.

- Specify the **-pp** option on the command line. See "Common Project Command Options" on page 16.
- Specify the entity argument on the command line.

Specifying Entities

To specify a project database for the **-pr** option or the PCLI_PR variable, you use the location of the project database, for example, `D:\producta`.

To specify an entity in a PCLI project command, you use the entity's path. There are two types of entity paths used by the PCLI:

- Project paths, which specify the projects and, optionally, the versioned files of the current project database.
- Workspace paths, which are either private or public.

Project Paths

The top-level of a project path is a forward slash (/), which specifies the current project database. Then, the hierarchy is as follows: /project/subproject/versionedfile. For example,
`/bridge/server/readme.txt` specifies the versioned file readme.txt in the server subproject of the bridge project.

NOTE For some commands, you may also specify revisions and revisions on a branch. For example, /bridge/server/readme.txt/1.2 specifies revision 1.2, while /bridge/server/readme.txt/1.2/1.2.1/1.2.1.0 specifies revision 1.2.1.0 on the 1.2.1 branch. See the instructions specific to each command.

When specifying a project for the entity argument in a command line, you can either specify the entire path or make the path relative to the current project. You specify the current project by either setting the PCLI_PP variable or specifying the **-pp** option on the command line. The entity argument for PCLI project commands is optional. You do not have to specify it if you have set a current project database and want the command to operate on the project database, or if you have set a current project and want the command to operate on that project. If you want the command to work on a project other than the current project or you want the command to work on a versioned file, you must specify a value for the entity argument.

For example, the syntax for the GetArchiveLocation is:

GetArchiveLocation [*options*] [*entity*]

You can use the following command line and not specify a value for the entity argument because you want the command to get the archive location for the current project, which is defined using the **-pp** option and a period (.) in this example.

```
GetArchiveLocation -prD:\producta -pp/bridge/server .
```

However, if you want to get the archive location for a versioned file, you would have to specify a value for the entity argument.

```
GetArchiveLocation -prD:\producta -pp/bridge/server readme.txt
```

NOTE To access versioned files at the 5.3/6.0 project level, you must use the syntax **/ProjectFolder**/*project_name*/[*ver_filename*].

/ProjectFolder is a keyword and must be typed exactly as it appears. It is *not* a place holder or part of an actual path on your system.

For example, **/ProjectFolder**/proj1/readme.txt, tells the PCLI to access the versioned file readme.txt in the 5.3/6.0 proj1 project (this versioned file is not in a 5.3/6.0 folder of the proj1 project). This syntax is typically used with the ListVersionedFiles command. For example:

```
pcli lvf -prD:\60prjdb -l -z /ProjectFolder/proj1/*.
```

You must specify the * to list the versioned files of a 5.3/6.0 project because versioned files are not typically directly beneath 5.3/6.0 projects but instead inside folders of 5.3/6.0 projects. Therefore, the **-z** option won't list versioned files of 5.3/6.0 projects as it would versioned files of 5.3/6.0 folders and projects.

Rules for specifying project paths

The following rules apply when you specify project paths for the entity argument:

- To specify the entire path of a project, the path must start with a forward slash (/). For example, /bridge/server specifies the server subproject of the bridge project.
- To specify a subproject of the current project, you need only use the subproject's name. For example, server specifies /bridge/server, assuming the current project is bridge.
- To specify the current project, you use one period (.). For example, . specifies /bridge, assuming the current project is bridge.
- To specify the parent of the current project, you use two periods (..). For example, .. specifies the project database in which the bridge project resides, assuming the current project is bridge.

The following rules apply when you specify an entity for the entity argument, the **-pp** option, or the PCLI_PP variable:

- Entity paths that contain spaces must be surrounded by single or double quotation marks. For example, "/project1/new subproject".

NOTE Simple quoting does not stop PCLI command-line splitting on spaces for most UNIX shells. This is only an issue when a file name or project name, for example, has embedded spaces. Quotation marks must be preceded by a backslash character (\). For example:

```
-pp\"/My project/\"
```

- Entity paths can contain wildcard characters, as follows:

Use this wildcard character...	To specify...
*	Any string. For example, *.c will match any entity path that ends with .c.
?	Any one character. For example, test??.java will match any entity path that starts with test, has any two characters after test, and ends with .java, such as test01.java, test02.java, etc.
[-]	A range of characters. For example, test.[a-d]* will match any entity path that has an extension that starts with the letters a, b, c, or d.
[*characters*]	Any characters within the square brackets. For example, test.[ch] will match either test.c, test.h, or both.
^	Any character but the one specified. For example, test.[^j]* will match all entity paths named test that have extensions that do not start with j.
\	The escape character. Place this character before any of the characters in this table to specify one of these special characters.

- If you surround an entity path with either single or double quotation marks, the wildcard characters are ignored; the entity path is not expanded.

Workspace Paths

In some PCLI project commands, you can optionally specify a workspace from which to take a value. For example, the workfile location for a project can vary depending on which workspace is set. Therefore, when you get a workfile location, you may want to specify from which workspace you want to get the value of the workfile location. The default is the user's default workspace.

You specify a workspace by either:

- Setting the PCLI_SP variable. See "Variables" on page 18.
- Specifying the **-sp** option on the command line. See "Common Project Command Options" on page 16.

You can specify either the Root Workspace, a private workspace, or a public workspace. To specify the Root Workspace, enter:

```
/@/RootWorkspace
```

To specify a private workspace, you use the following syntax:

`/@/`*`userID`*`/`*`parent_workspace/`*`[`*`child_workspace`*`]`

where *userID* is your user ID and is case sensitive. For example, `/@/AdamJ/myprivateworkspaces/mylocaldrive`.

To specify a public workspace, you use the following syntax:

`/@/Public/`*`parent_workspace/`*`[`*`child_workspace`*`]`

For example, `/@/Public/networkdrive`. In this example, there is no child workspace specified.

Command Options

Command options are values that you provide to a PCLI command to tell the command the action to take. For example, through the use of options, you can tell the ListProjectDB command to list only the current project database or to list all active project databases.

Command options have one of two forms:

- A flag, which starts with a hyphen (-), often followed by a value. Flags are case insensitive.
- A positional value (argument) in the command line.

For example:

```
pcli addfiles -prD:\producta F:\dev\work
              /               /
          A flag          A positional value
```

The **-h** flag is available for all commands. This flag displays the online help for a specific command (`pcli` *`command_name`* **`-h`**) or displays a list of available PCLI commands (`pcli` **`-h`**).

You can suppress the banner output of PCLI via the option -nb (No Banner). This option should be specified in-between pcli and the command PCLI should execute. For example, to run the AddFiles command without banner output, you would execute:

```
pcli -nb addfiles ...
```

Additionally, both PCLI and CLI can suppress their banner output if the environment variable PVCS_NO_BANNER exists with a value of true. When PCLI is called with the **-nb** option, it automatically sets the PVCS_NO_BANNER variable to true for all child processes, so any external PCLI or CLI command getting launched from a PCLI script using **run -e** or via an Event Trigger will automatically get its banner suppressed too.

Common Project Command Options

The following project command options are available for most of the PCLI project commands.

Project Command Option	Definition
`-pr`*`projectdatabase`*	Specifies the project database to use
`-id`*`userID`*`[:`*`password`*`]`	Specifies a user ID and password (if one is defined) to use
`-pp`*`project`*	Specifies the current project
`-sp`*`workspace`*	Specifies the workspace to use

Let's take a closer look at these common project command options.

-id — This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN — If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp — This option can be specified on the command line to define the current project (the project to operate on). If the PCLI_PP variable (see Figure 1-2, "Reserved Variables Used to Supply Values," on page 20) is set, **-pp** overrides the value of the PCLI_PP variable for a single command execution. This option is optional for all project commands. If a current project is not defined, then project database entities must be specified using their fully qualified entity paths, for example:

```
          /bridge/server
            /       /
A project          A subproject
```

The entity path is relative to the root of the project database. For more information on entity paths, see "Specifying Entities" on page 13.

-pr — This option can be specified on the command line to define which project database to work with. If the PCLI_PR variable (see Figure 1-2, "Reserved Variables Used to Supply Values," on page 20) is set, **-pr** overrides the value of the PCLI_PR variable for a single command execution. Most project commands require that a project database be specified.

-sp — This option can be specified on the command line to define the workspace to use. If the PCLI_SP variable (see Figure 1-2, "Reserved Variables Used to Supply Values," on page 20) is set, **-sp** overrides the value of the PCLI_SP variable for a single command execution. If a workspace is not specified, then the user's default workspace is used.

Values for Options

Besides an actual value for an option (as previously shown), a value specified in a command line can take the form of:

- A variable
- Command output
- The name of a list file

The PCLI command interpreter translates these forms of input into actual values that the command can use.

Variables

Using the PCLI, you can define variables within a script to provide values throughout the execution of the script. Three types of user-defined variables are supported:

- Typical string variables
- Sequential array variables
- Associative array variables

See the "Set command" on page 142 for more information about variables.

In addition, the PCLI has a set of reserved variables. These variables have four different functions:

- Variables that you can set to supply a command with values. See the table on Figure 1-2, "Reserved Variables Used to Supply Values," on page 20.
- Variables that are set by PCLI for your reference. See Figure 1-3, "Reserved Variables Set by PCLI for Reference," on page 20.
- A variable that you set to define the delimiter characters the PCLI uses to divide the command line into separate words. This variable is PCLI_IFS. See "Word Splitting" on page 22.
- A debugging variable that you set to define the script trace mode that is in effect, if any. This variable is PCLI_SCRIPT_TRACE. See "Debugging Scripts with PCLI_SCRIPT_TRACE" on page 28.

Setting Variables

You can set the value of PCLI variables in PCLI scripts by using the PCLI Set command. For example, in a PCLI script:

```
Set -vPCLI_PR D:\producta
getconfigfile /bridge
```

or

```
Set PCLI_PR=D:\producta
getconfigfile /bridge
```

In either case, the variable is global within the script. See the "Set command" on page 142 for more information.

In addition, you may find it useful to make a PCLI reserved variable an operating system environment variable by using the appropriate operating system command. Platform specific examples include:

- Windows:
  `Set PCLI_PR=D:\producta`
- UNIX Bourne Shell (sh):
  `PCLI_PR=/usr/serena/products; export PCLI_PR`
- UNIX Korn Shell (ksh) or Bourne Again Shell (bash):
  `export PCLI_PR=/usr/serena/producta`
- UNIX C Shell (csh):
  `setenv PCLI_PR /usr/serena/producta`

In this case, the variable is global to all PCLI scripts, commands executed directly, and commands executed interactively during the current operating session. All environment variables are converted to PCLI variables and are available to scripts.

Naming Variables

Variable names can contain letters, numbers, and the underscore character (_). All variables, except for variables used for positional parameters to PCLI functions and scripts, must start with a letter or the underscore character. Variables that consist of only numbers are positional parameters to PCLI functions and scripts, and are set by PCLI when a script or function is executed.

IMPORTANT! Variables are case sensitive, so $VAR is different from $Var which is different from $var.

Referencing Variables

You reference a typical string variable by placing either ${*variable_name*} or $*variable_name* in the appropriate position in the command line. For example, from a script, the second line references the variable that is set in the first line:

```
set -vconfig $(getconfigfile /bridge)
setconfigfile -c${config} /newprj
```

You reference an array variable by placing either ${*variable_name*[*index*]} or $*variable_name*[*index*] in the appropriate position in the command line. If you do not specify a value for *index*, all of the values of the array are returned. The value for *index* is an integer for a sequential array or a string for an associative array. For example, to return all of the values of an array variable:

```
run -e get -l $FileList[]
```

Or, to return only a single value from an array, you must specify an element of the array:

```
echo $FileList[1]
```

NOTE When referencing an element of an array, you must not have leading or trailing spaces around the name of the element. For example, you cannot have:

```
echo $FileList[ 1 ]
```

Reserved Variables

Table 1-2 defines the PCLI reserved variables that are used to supply a command with values.

Table 1-2. Reserved Variables Used to Supply Values

This reserved variable. . .	Contains the value of the. . .
PCLI_ID	Current user ID and password (optional) **NOTE** For information on using EventPassword for PCLI authentication, see "Authenticating PCLI Users with EventPassword" on page 22.
PCLI_IFS	Delimiter characters. The default values are a space, tab, carriage return, and new line. For more information, see "Word Splitting" on page 22.
PCLI_PP	Current project
PCLI_PR	Current project database
PCLI_SP	Current workspace

Once you have set these variables, you do not need to reference them in a command line because the PCLI automatically substitutes the values in the command line. For example, the GetConfigFile command requires a value for a project database such as:

```
pcli getconfigfile -prD:\producta /bridge
```

If you set a value for the PCLI_PR variable, you need not specify the **-pr** option for the GetConfigFile command because the PCLI will use the value of the PCLI_PR variable. You need only enter:

```
pcli getconfigfile /bridge
```

In addition, there is a set of PCLI reserved variables that are set by PCLI (see Table 1-3). These variables are not to be set by you; they are provided for reference.

Table 1-3. Reserved Variables Set by PCLI for Reference

This reserved variable. . .	Contains the value of the. . .
#	Number of positional parameters (arguments).
0	Executed script file name.
number	Positional parameter (argument).
*	Positional parameters, $1 $2 $3 . . .
?	Status of the last command executed (return code).
PCLI_LINENO	Line number of the current command in the script file. If the current command was not read from a script, then this variable is not defined.
PCLI_SCRIPT_FILE	File name of the script the current command was found in. If the current command was not read from a script, then this variable is not defined.
PCLI_VERSION	Version number, including the build number, of the Version Manager PCLI install you are running.

You can use the PCLI_VERSION variable from the command-line and from scripts.

Examples **Example 1:** The following example shows how to access PCLI_VERSION from the command-line:

```
pcli echo "PCLI_VERSION=$PCLI_VERSION"
PCLI_VERSION=6.8.00 (Build 055)
```

Example 2: The following example shows how to use PCLI_VERSION in a script that verifies the version number before proceeding:

```
if [ "$PCLI_VERSION" < "6.8.00" ]
{
   # NOTE: Avoid testing "$PCLI_VERSION" = "6.8.00"
   # since the $PCLI_VERSION string also includes the
   # build number, and "6.8.00 (Build 055)" != "6.8.00".
   # As such, use operators like <, <=, > and >= to take
   # this into account (use alphabetic string comparison).
   echo "This script requires Version Manager 6.8.00 or newer."
       exit 1
}
# Version Manager 6.8-specific commands to follow
DeleteUser -idAdmin:admin Joe
# Etc...
```

Command Output

Command output can be inserted as a value for a command option in a command line. The syntax for the use of command output is:

$(*command*)

or

$[*command*]

where *command* is a complete command including options. The command is executed and its output is inserted in the command line.

For example, you could get the name of the configuration of one project and associate it with another project:

```
pcli setconfigfile -c$(getconfigfile /bridge) /newprj
```

This example tells the PCLI command interpreter to get the value of the configuration file associated with the bridge project and associate that configuration file with the newprj. In this example, the two projects have the same project database because no project databases are specified on the command line.

The result of command substitution is split into words unless you use the $[*command*] syntax. In this case, the command substitution is split on each new line. The $[*command*] syntax makes it easier to use commands that output one path per line where the path may contain spaces.

List Files

A list file contains command-line input for PCLI commands. When used in a command line, the contents of the file are substituted for the value of the option in which the file was specified. The syntax for the use of a list file is:

@*list_file*

You can use a list file to perform an operation on multiple files in a project. For example, you could simplify the task of importing versioned files into a new project by creating a list file containing the locations of the versioned files in the old project. You could first direct the output of the ListVersionedFiles command to a list file, and then use this list file when executing the ImportArchives command:

```
pcli ListVersionedFiles -prD:\producta -z -aw /bridge > list.tmp
pcli ImportArchives -prD:\productb -pp/newprj @list.tmp
```

Authenticating PCLI Users with EventPassword

You can use the __EventPassword__ command-line macro or the EVENTPASSWORD environment variable to authenticate PCLI users. However, by default, this feature is disabled and any such attempt will fail.

Enabling EventPassword for PCLI

To enable EventPassword for use with PCLI:

1. Open the following file in a text editor:

   Install_Dir\vm\common\pvcsprop\pvcs\vm\security.properties

2. Add the following line:

   vm.allowEventPassword=enabled

3. Save the file and repeat the above steps on every system on which you want to enable authentication with EventPassword.

Using EventPassword with PCLI

Once you have enabled EventPassword for use with PCLI, you can use it to authenticate PCLI users by setting the PCLI_ID variable or by passing arguments to the PCLI script. For example:

```
set PCLI_ID="$EVENTUSERID:$EVENTPASSWORD"
```

or

```
-id"__EventUserID__:__EventPassword__"
```

For more information on EventUserID and EventPassword, see the *Version Manager Command-Line Reference Guide*.

Word Splitting

The first step in command line processing is called word splitting, which is the process of dividing the command line into separate words based on delimiter characters. The default delimiter characters for PCLI direct command execution are a space, tab, carriage return, and new line.

Using Quotation Marks

The effect of the delimiter characters can be modified by the use of quotation marks. Single and double quotation marks suspend word splitting. For example, to use spaces in a path you can use quotation marks in the command:

```
pcli getarchivelocation -pr"D:\My Projects\producta"
```

Without quoting, the path would be broken at the space and the command would not behave as desired.

Word splitting is also performed on the result of:

- List file input processing
- Variable expansion
- Command output substitution

In these cases, you can define the PCLI_IFS reserved variable to define the characters you want to use as delimiter characters. The default values for this variable are a space, tab, carriage return, and new line. See the **-r** option of the "Set command" on page 142 for how to set the PCLI_IFS variable.

Quoting Rules

The following list provides rules for quoting in the PCLI:

- Single quotation marks can be used to escape double quotation marks.
- Double quotation marks can be used to escape single quotation marks.
- A backslash can be used to escape either single or double quotation marks only.
- A backslash preceding any character other than a quotation mark has no special meaning.
- A backslash has no special effect inside a quoted area, it is just a character.
- Variable expansion does not happen inside of single quotation marks.
- Double quotation marks around a variable or a command output definition prevent word splitting of the results of variable expansions and command output substitutions.
- Command output substitution does not happen inside of single quotation marks.
- @file expansion by the PCLI interpreter does not happen within either single or double quotation marks.
- Entity path expansion does not happen within single or double quotation marks.
- All commands including the Run and Set commands will remove all unquoted quotation marks from arguments after variable expansion and command output substitution and before execution.
- A command output substitution has an implied Run command as part of its definition.
- When a quoted area starts with single or double quotation mark, the only character that can end the quoted area is another quotation mark of the same kind.
- Quoted areas can extend into the next line. The newline characters become part of the quoted string.

Quoting Tips

This section outlines a few of the situations where quoting is useful and notes some of the pitfalls to watch out for when using quotation marks.

Quotation Marks in Paths

Quotation marks that are contained inside of a file specification should be avoided. For example:

```
C:\"Program Files"\somedir\anotherdir\somefile
```

The first backslash will be removed when this is an argument to a command because the backslash escapes the quotation mark.

If the above example is part of a script, enclosing it in single quotation marks will preserve the backslash. (Proper syntax for use at a command prompt depends upon the shell you are using. See "Passing Quotation Marks Through a Shell" on page 26.)

Special PCLI Characters in Paths

If any of your paths, files, or other data to be processed by a PCLI script contain special PCLI characters, like $, -, @, etc., you must quote them so that they will not be interpreted by a PCLI command. Place single quotation marks around the word that contains these characters.

Special Shell Characters in Commands

When issuing PCLI commands from the command-line, you must quote any PCLI command arguments that have special meaning to your shell. For example:

```
pcli run -">"dir.txt -e dir
```

—or—

```
pcli run "->dir.txt" -e dir
```

Without the quotation marks, your shell would interpret the angle bracket as a redirection flag instead of leaving that to the Run command.

Command Shells and Quotation Marks

Before using quotation-mark bearing PCLI commands from a command prompt, you must consider how your operating system interprets quotation marks. Windows, for example, interprets double quotation marks (") differently than single quotation marks ('), as the following examples illustrate.

Examples

Example 1: The following example shows how double quotation marks are interpreted from the Windows command prompt. The command:

```
pcli echo -ns "Three  spaces and quotes."
```

Passes three arguments to PCLI:

- `echo`

- `-ns`
- `Three  spaces and quotes.`

Which results in the output:

```
Three  spaces and quotes.
```

The double quotation marks served to bundle the space separated words into a single argument, preserving the spaces. The quote pair was removed by the Windows shell before PCLI interpretation.

Note if you run the same Echo command inside a script, the quotation marks will be included in the output:

```
"Three  spaces and quotes."
```

Example 2: The following example shows how single quotation marks are interpreted from the Windows command prompt. The command:

```
pcli echo -ns 'Three  spaces and quotes.'
```

Passes six arguments to PCLI:

- `echo`
- `-ns`
- `'Three`
- `spaces`
- `and`
- `quotes.'`

Which results in the output:

```
'Three spaces and quotes.'
```

In Windows, single quotation marks are just another character, so the spaces break the single-quoted string into four separate arguments. The three spaces between the words `Three` and `spaces` are discarded. The Echo command then concatenates the arguments and reforms the sentence with a single space between each word. The **`-ns`** option allows the quotation marks to pass through the PCLI interpreter.

Note if you run the command inside a script, the output will retain the three spaces between the words `Three` and `spaces`:

```
'Three  spaces and quotes.'
```

Passing Quotation Marks through a Shell or Command

You must quote, or escape, quotation marks when you want to prevent an operating system shell or PCLI command from interpreting quotation marks and instead pass them along unaltered to be displayed or used by another command. PCLI is compatible with the

following methods of escaping quotation marks (though your command-line shell may not be):

To escape...	Use...	For example...
Double quotation marks (")	Single quotation marks (') or backslashes (\).	She said, '"You can quote me."' or She said, \"You can quote me.\"
Single quotation marks (')	Double quotation marks (") or backslashes (\).	She said, "'You can quote me.'" or She said, \'You can quote me.\'

Thus, a command of the form:

```
pcli echo -ns She said, \"You can quote me.\"
```

Produces this result:

```
She said, "You can quote me."
```

NOTE A backslash that is itself inside of unescaped quotation marks is treated as a normal character and cannot serve to escape quotation marks.

Passing Quotation Marks Through a Shell

When executing PCLI commands from a command shell that interprets quotation marks as special characters, you must quote or escape any quotation marks that you want to pass through to the PCLI interpreter. In doing this, you must consider how your shell interprets double quotation marks ("), single quotation marks ('), and backslashes (\). Windows, for example, does not interpret single quotation marks as special characters, so you cannot use them to quote double quotation marks.

Example

The following example shows how to use backslashes from the Windows command prompt to escape double quotation marks. The command:

```
pcli echo -ns \"Three  spaces and quotes.\"
```

Results in:

```
"Three  spaces and quotes."
```

Passing Quotation Marks through a PCLI Command

You can prevent a PCLI command from interpreting quotation marks and instead pass them along unaltered to be displayed or used by another command. This is done by quoting or escaping the quotation marks you wish to pass through.

If you wish to accomplish this from the command-line rather than from within a script, you must also consider how your system shell interprets quotation marks.

Match Quotation Marks Carefully

Be careful to correctly match quotation marks. Unmatched quotation marks will cause the parsing code to continue a quoted string into the next line. Commands will not parse properly from that point forward.

Quoting Examples

Example 1

```
#Suppose you want to execute the external command vdel and
#pass to it the contents of a file. The @file should not be
#interpreted by the PCLI script, because the expanded list
#of arguments might be too long to fit in the operating
#system's command buffer (where the external command will
#be executed).
run -ns -e vdel -r1.0 "@file"
```

or

```
run -ns -e vdel -r1.0 '@file'
```

Example 2

```
#Suppose you want to use filenames or paths in a script that
#have characters in them that are special characters to the
#PCLI interpreter, like $.
get 'foo$3.class'
#Suppose you want to use filenames or paths that have
#characters in them that are special characters to a shell,
#like $.
run -ns -e vdel -r1.0 'foo$3.class'
```

Example 3

```
# Suppose you want to add double quotation marks around a
# variable's contents.
set -vFOO '"'"$FOO"'"'
```

or

```
set -vFOO \""$FOO"\"
```

or

```
set -ns -vFOO "$FOO"
```

Example 4

```
#Promote the versioned files in the 'bridge' project using
#the same list of configuration files that the desktop client reads.
set -a -vConfigList \
   $[gcf -pr"D:\Program Files\Serena\vm\pcli_sample" -a /bridge]
set -vi 0
While Test $i < $(arraysize ConfigList)
{
   set -i$i -vCmdArgs '-c"'"$ConfigList[$i]"\"
   calc -vi $i + 1
}
run ->listfile.txt \
   lvf -pr"D:\Program Files\Serena\vm\pcli_sample" -aw /bridge
run -ns -e vpromote $CmdArgs[] -g"QA" "@listfile.txt"
```

In the first Set command, the file path of the sample database is quoted to prevent word splitting at the space character in `program files`, thus maintaining the entire file path as one unit.

The While loop double quotes each element of the ConfigList array and places a **-c** in front of each element. The objective is to place quotation marks around each element that

is used by the external vpromote command, which is executed in the last line of the script:

- The double quotation marks ensure that the vpromote command will handle the space in the configuration file paths correctly.
- The single quotation marks around the **-c** are removed by the Set command and prevent the **-c** option, which is to be used by the vpromote command, from being interpreted by the Set command.
- The pair of double quotation marks surrounding `$ConfigLIst[$i]` is required because the value of this variable may contain spaces and we want to retain these spaces in the variable that is being assigned by Set.

The **-ns** option used by the Run command prevents the quotation marks surrounding the configuration file paths from being stripped by the Run command, thus leaving the quoted file paths intact for the vpromote command.

Debugging Scripts with PCLI_SCRIPT_TRACE

The PCLI_SCRIPT_TRACE variable allows you to enable three different script trace debugging modes:

Mode	Syntax	Shows
0	**set** PCLI_SCRIPT_TRACE=0	Normal command output. (This turns off the script trace feature.)
1	**set** PCLI_SCRIPT_TRACE=1	Each line before variables are expanded.
2	**set** PCLI_SCRIPT_TRACE=2	Each line after variables are expanded.
3	**set** PCLI_SCRIPT_TRACE=3	Each line before and after variables are expanded.

Debug information will be shown under STDERR (standard error) for each line that occurs after the PCLI_SCRIPT_TRACE variable is set to `1`, `2`, or `3`. Each line is prefaced with a `1` or `2` to indicate if it represents the pre or post variable expansion state. Output will return to normal for all lines that occur after the variable is set to `0`.

Mode Examples

The examples below use the following script, which is shown here with its normal (mode `0`) output:

```
C:\>type Example.pcli
set extra=blue
set colors=red green $extra
for color in $colors
{
   echo "color=$color"
}

C:\>pcli run -sExample.pcli
```

```
PVCS Version Manager PVCS Version Manager (PCLI) v8.6.1.0 (Build 162)
    for Windows/x86_64
Copyright (C) 1985-2018 Serena Software, Inc., a Micro Focus company.
    All rights reserved.

color=red
color=green
color=blue
```

Mode 1 Example

```
C:\>set PCLI_SCRIPT_TRACE=1
C:\>pcli run -sExample.pcli
PVCS Version Manager (PCLI) v8.6.1.0 (Build 162) for Windows/x86_64
Copyright (C) 1985-2018 Serena Software, Inc., a Micro Focus company.
    All rights reserved.
+1 Example.pcli[1]: set extra=blue
+1 Example.pcli[2]: set colors=red green $extra
+1 Example.pcli[3]: for color in $colors
+1 Example.pcli[5]: echo "color=$color"
color=red
+1 Example.pcli[5]: echo "color=$color"
color=green
+1 Example.pcli[5]: echo "color=$color"
color=blue
```

Mode 2 Example

```
C:\>set PCLI_SCRIPT_TRACE=2
C:\>pcli run -sExample.pcli
PVCS Version Manager (PCLI) v8.6.1.0 (Build 162) for Windows/x86_64
Copyright (C) 1985-2018 Serena Software, Inc., a Micro Focus company.
    All rights reserved.
+2 Example.pcli[1]: set extra=blue
+2 Example.pcli[2]: set colors=red green blue
+2 Example.pcli[3]: for color in red green blue
+2 Example.pcli[5]: echo "color=red"
color=red
+2 Example.pcli[5]: echo "color=green"
color=green
+2 Example.pcli[5]: echo "color=blue"
color=blue
```

Mode 3 Example

```
C:\>set PCLI_SCRIPT_TRACE=3
C:\>pcli run -sExample.pcli
PVCS Version Manager (PCLI) v8.6.1.0 (Build 162) for Windows/x86_64
Copyright (C) 1985-2018 Serena Software, Inc., a Micro Focus company.
    All rights reserved.
+1 Example.pcli[1]: set extra=blue
+2 Example.pcli[1]: set extra=blue
+1 Example.pcli[2]: set colors=red green $extra
+2 Example.pcli[2]: set colors=red green blue
+1 Example.pcli[3]: for color in $colors
+2 Example.pcli[3]: for color in red green blue
```

```
+1 Example.pcli[5]: echo "color=$color"
+2 Example.pcli[5]: echo "color=red"
color=red
+1 Example.pcli[5]: echo "color=$color"
+2 Example.pcli[5]: echo "color=green"
color=green
+1 Example.pcli[5]: echo "color=$color"
+2 Example.pcli[5]: echo "color=blue"
color=blue
```

Mid Script On and Off Example

The following script shows how script tracing can be turned on and off at various points within a script.

```
C:\>type Example2.pcli
set color=Blue
echo Red
set PCLI_SCRIPT_TRACE=3
echo $color
set PCLI_SCRIPT_TRACE=0
echo Green

C:\>pcli run -sExample2.pcli
PVCS Version Manager (PCLI) v8.6.1.0 (Build 162) for Windows/x86_64
Copyright (C) 1985-2018 Serena Software, Inc., a Micro Focus company.
    All rights reserved.
Red
+1 Example2.pcli[4]: echo $color
+2 Example2.pcli[4]: echo Blue
Blue
+1 Example2.pcli[5]: set PCLI_SCRIPT_TRACE=0
+2 Example2.pcli[5]: set PCLI_SCRIPT_TRACE=0
Green
```

Date and Time Formats

Many PCLI commands accept a date and time attribute. The expected format of this attribute is determined by the regional settings of the system you are using. Here are examples for several regions.

US formats — Valid examples for the United States include:

```
12/30/01
12/30/2001 1:05:20 PM
Dec 30, 2001 1:05PM
Dec 30, 2001 13:05
December 30, 2001
```

Dutch formats — Valid examples for the Netherlands inlude:

```
30-12-01
30-12-2001 1:05 PM
```

```
30-Dec-2001 1:05PM
30-Dec-2001 13:05
30 December 2001
```

UK formats Valid examples for the United Kingdom include:

```
30/12/01
30/12/2001 1:05:20 PM
30-Dec-2001 1:05PM
30-Dec-2001 13:05
30 December 2001
```

If you enter an invalid date and time value (such as TEST), PCLI will display examples compatible with the regional settings in effect on your system.

IMPORTANT! If the date and time attribute contains spaces, you must enclose the attribute in quotation marks. For example, `"Dec 30, 2001"`.

PCLI Functions

A function is a piece of PCLI code that is defined once in a script and can be invoked many times. PCLI functions can be passed arguments that specify the value or values that the function is to operate on. PCLI functions are defined as follows:

- The name of the function
- The PCLI commands that comprise the body of the function, contained within curly braces
- Arguments to the commands are represented by positional parameters, $1, $2, etc.

```
function_name()
{
         PCLI command
         PCLI command
         PCLI command. . .
}
```

For example:

```
checkout()
{
   run ->files.tmp listversionedfiles -pr$1 -z -aw /$2
   run -ns -e get -l '@files.tmp'
}
```

PCLI functions can be nested, which means that you can define functions inside of functions. In this case, the nested functions can only be called inside the function in which they are defined.

Using Functions

Functions are an efficient way to reuse code for actions that you frequently perform, such as checking in and out files.

Once you define a function in a script, you invoke it within the script in one of the following ways:

- By following the function's name with a list of arguments. For example:
  `checkout D:\producta bridge`
- By using the Run command. See the description of the Run command on 139. For example:
  `run -f``checkout D:\producta bridge`

One way to reuse PCLI functions among scripts is to create a file that contains only a library of PCLI functions, and then include this file within any script by placing `@`*`function_file`* on a line by itself within the script. For more information see the "@ command" on page 45.

Chapter 2
Scripting

Introduction

The primary reason for the release of the PCLI is to provide you with the ability to write scripts to perform multiple actions on projects by simply invoking scripts.

Scripts can be used to perform a variety of tasks for you, such as:

- List all of the versioned files in a project and then check them out.
- Add a project to a project database and then add workfiles to the new project.
- Create a new project database
- Get the current project database location
- Get a project's configuration file
- Set a configuration file for a project
- Get a project's workfile location
- Set a workfile location for a project

You can use the PCLI commands in combination with any scripting language supported by your platform. For example, UNIX shell scripts, Windows batch scripts (.BAT), and Configuration Builder scripts.

You can also create PCLI scripts, scripts that contain only PCLI commands and functions. The PCLI commands that are provided specifically for scripting purposes are:

- @
- Echo
- If
- Elif
- Else
- For
- Set
- Return
- Exit
- Run
- While
- Calc
- ArraySize
- Test
- Break
- Continue
- Readline

Refer to Chapter 3, "PCLI Command Reference" on page 43 for complete information about these commands.

Executing Scripts

The method that you use to run scripts depends on the type of script you have written. To run PCLI scripts, scripts that contain only PCLI commands, you use the PCLI Run command. To run external scripts, you can execute the script from the command prompt.

To run a PCLI script, you enter:

`pcli run -s`*`scriptfile`* [*`arguments`*]

PCLI scripts can be passed arguments that specify the value or values that the script is to operate on. For example:

```
pcli run -sscript.pcli D:\producta /proj1
```

And, the script looks like this:

```
# Set the first argument passed to the variable PDB
# The variable is quoted in case the value passed
# contains spaces
set -vPDB "$1"
# Set the second argument passed to the variable PROJECT
set -vPROJECT "$2"
# Use the values of the variables in the command line
getconfigfile -pr"$PDB" -pp"$PROJECT"
```

If the values you pass to the script have spaces in them, you must quote them on the command line. For example:

```
pcli run -sscript.pcli "D:\new product" /proj1
```

Syntax of PCLI Scripts

When writing PCLI scripts—scripts that contain only PCLI commands—you can:

- Enter comments in the script by using the pound character (#) as the first character in the line or the # on the same line as the script code as long as the # is separated from the rest of the line by at least one space or tab and is the first character of a word.
- Continue a line by entering a backward slash (\) at the end of the line. The \ must be separated from the rest of the line by a space or tab and be the last character on the line. Otherwise, PCLI interprets the \ as part of the text in the line.
- Use any PCLI command in the script but do *not* precede the command with the word pcli.
- Execute commands other than PCLI commands (such as operating system commands) in the script by using the PCLI Run command. For example:

  ```
  run -e cp files myfiles
  ```

  executes the UNIX Copy command.

NOTE On Windows systems when you Run a batch file, you must specify the .bat file extension.

- Define PCLI functions in the script. See "PCLI Functions" on page 31.

User IDs

When you run a PCLI script, there is one root login for each user ID logged in to a project database. Each root login has a current workspace associated with it. The default user ID

is the first user ID logged in to the project database. If the PCLI script is being run from the desktop client, the default user ID may have been set via the desktop client. A PCLI command that is not associated with a user ID is assigned to the default user ID.

For example, a PCLI script may open a project database (PDB-A) using a user ID (U1), assigning a workspace (W1). A subsequent PCLI command in the script may reference PDB-A using a different user ID (U2), assigning a second workspace (W2). A subsequent PCLI command that references PDB-A without specifying a user ID (using either the **-id** parameter or a local PCLI_ID variable) will be assigned the root login of the first user with the first workspace (U1 and W1). If the script is multi-threaded and several threads are sharing a root login, then one thread changing the workspace will result in the workspace being changed for all threads using that root login.

Using the PCLI with the Existing Command-Line Interface in Scripts

This sections discusses information that is important for you to know when you use PCLI commands along with existing command-line interface commands in PCLI scripts.

NOTE Many CLI commands, such as Get and Put shown in the example below, now have PCLI counterparts. Unless your script also needs to run on PCLI releases older than 6.7.00, we recommended that you use the PCLI version of the Get and Put commands, as they are project and workspace aware.

UNC Paths

Often you will find it useful to output a list of versioned files using the PCLI ListVersionedFiles command that can be used with the existing command-line interface Get, Put, and Vcs commands. The **-aw** option of the ListVersionedFiles command lists the output in a format that can be used by the Get, Put, and Vcs commands. That format is archive path(work path), for example:

```
"D:\producta\archives\bridge\resrc.h-arc(D:\producta\work\bridge\resrc.h)"
```

The output of the ListVersionedFiles command using the **-aw** option surrounds the output with double quotation marks so that if a space exists in any of the path names, the existing command-line interface commands will operate correctly on the path names. See "ListVersionedFiles command" on page 122.

If you are using UNC paths, the output of the ListVersionedFiles command using the **-aw** option does not work with the existing command-line interface commands because the existing command-line interface requires an extra backslash character (\) in front of the two backslash characters that begin a UNC path **when** the path is inside quotation marks.

The following script adds the required extra backslash character to the archive and work paths so that the UNC paths can be used with the existing command-line interface commands.

```
# Defined the project database location to use.
set -vPCLI_PR "\\server\pdbs\SampleDb"

# Define the top level project to use (/ = ALL).
```

```
set -vPCLI_PP "/bridge"

# Loop through all versioned files at and below the top level project.
set -vi 0
for file in $[lvf -l -z]
{
   # Obtain the archive location for the versioned file and
   # add an extra backslash if it is defined as a UNC path.
   set -l -ns -vArchive  $[-ns GetArchiveLocation -pp/ \
            "$file"]
   if test "$Archive" = "\\*"
   {
            set -vArchive "\$Archive"
   }

   # Obtain the workfile location for the versioned file and
   # add an extra backslash if it is defined as a UNC path.
   set -l -ns -vWorkfile $[-ns GetWorkLocation -pp/ \
            "$file"]
   if test "$Workfile" = "\\*"
   {
            set -vWorkfile "\$Workfile"
   }

   # Place this versioned file in the array GetList, while
   # adding quotation marks around the entire string and
   # parentheses around the workfile location.
   set -a -i$i -vGetList '"'"$Archive($Workfile)"'"'
   calc -vi $i+1
}

# Pass the GetList array to the external command "get".
# We do this by first writing the array out to a temporary
# file. Although we could pass the array straight to the
# command "get" using "run -e get $GetList[]", this could
# overflow the OS command line buffer if the project is
# large enough.
set -vTempFile "filelist.txt"
run -ns ->$TempFile echo -ns $GetList[]

# Now run the external command "get" and pass it the name of
# the file that contains the list of archive and workfile
# locations.
run -ns -e get -l "@$TempFile"
```

Example Scripts

Example 1: Uses the output of the List command to get any files with names matching `*.cpp`, `*.h`, or `*.java` from the projects /bridge and /chess/client.

```
# Project Database
set -vPCLI_PR "C:\Users\All Users\Serena\VM\SampleDB"
```

```
# UserID + Password
set -vPCLI_ID "admin"

# Unset PCLI_PP to avoid conflicts with quoted fully
# qualified entity paths.
# NOTE:  This issue is resolved as of Version Manager 6.8.00 # Service
    Pack 3 (Build d128.03)
set -vPCLI_PP

# Last argument(s) in the "list" command below are the
# top-level projects from which the files should be fetched
# recursively down. Projects are specified as PCLI entity
# paths, following the format /<project>/<subproject>/...
#
# Example entity paths:
#
# "/"   Entire PDB
# "/bridge" Project "bridge" (under PDB root)
# "/chess/client" Project "client", located under
# project "chess"
#
#
# Run the "list" command and capture its output one line
# at a time using "$[]"
# Use this to feed a "for" loop, such that the variable
# "file" will be assigned one line of output from the list
# command for every itteration of the loop, assigning it a
# fully qualified entity path to versioned files from the
# projects /bridge and /chess/client with names matching
# *.cpp, *.h or *.java
#
for file in $[-ns list -tVersionedFile -l -z -m"*.cpp" \
   -m"*.h" -m"*.java" "/bridge" "/chess/client"]
{
   # Get files into workfile location specified in
   # the active workspace get "$file"

   # Get files into directory "D:\build". -bp/ states that
   # the target directory is at the hierarchical level of
   # the PDB root, so a subdirectory is created for every
   # subproject under the PDB root.
   get -o -a"D:\build" -bp/ "$file"
}
```

You would execute this script by using the Run command. For example:

pcli **run -s**checkout.pcli

Example 2: List all of the versioned files in a project and then check in only the files that have changed. You may find it useful to have a script like this to execute at the end of each day so that you can quickly check in your work before you leave for the day.

As in Example 1, the ListVersionedFiles command uses the **-aw** and **-z** options. Notice in this example the Put command uses the **-n** option, which tells Version Manager to answer no to queries issued by Put. Version Manager will answer no to the question, "The workfile unchanged. Check in anyway?". Therefore, files that are unchanged will not be checked in. Also, the Put command uses the **-m**@*file_name* option to read the change description

for the workfiles from a file. This way you can enter a change description for the workfiles into one file that is used by Version Manager. For example, you could enter "Updates made in response to issue #4508."

This example uses positional arguments.

```
# Command usage: pcli run -scheckin.pcli <PDB> <PROJECT>
# Assign the first positional argument to the PCLI variable
# PDB and set the second positional argument to PROJECT
set -vPDB "$1"
set -vPROJECT "$2"
Echo Listing the versioned files.
run ->files.tmp listversionedfiles -z -aw -pr"$PDB" \
    -pp"$PROJECT"
Echo Checking in the files.
# Must use the PCLI Run command to execute Put
# Because Put is not a PCLI command
run -ns -e put -n '-m@descript' '@files.tmp'
Echo Deleting the listfile.
# Same goes for the DOS command, del
run -e cmd /c del files.tmp
```

You would execute this script by using the Run command. For example:

```
pcli run -scheckin.pcli "D:\demo product" /bridge
```

In some cases, you may have projects that have configuration files associated with them. In these cases, you may want to specify the configuration file(s) associated with the project on the command so that the PCLI operates according to the definitions in the configuration file(s). See the For command example on 76 for an explanation of how to get the configuration files associated with a project and specify the values on the command line.

NOTE You can use this technique of listing all versioned files in a project and performing an action on the listed files for several different reasons, such as applying a versioned label to the files, locking the files, unlocking the files, etc.

For example, to apply a version label to the listed files, you would replace the Put command with the VCS command in the script as follows:

```
run -ns -e vcs -v"Release" '@files.tmp'
```

Example 3: Add a project to a project database and then add workfiles to the new project.

This example uses the Set command to set the PCLI_PR variable. Therefore, the project database does not have to be specified in the CreateProject and AddFiles commands.

```
# Command usage: pcli run -saddproj.pcli <PDB>
# Assign the PDB argument to the PCLI_PR variable
set -vPCLI_PR "$1"
createproject -w"$PCLI_PR"\work\newprj newprj
addfiles -pp/newprj -c -t"new files" C:\work\prj1\*.*
```

You would execute this script by using the Run command. For example:

```
pcli run -saddproj.pcli "D:\new product"
```

Example 4: Create a project database, create a project in the database, import archives into the project, assign version labels to some versioned files, set the workfile locations for some projects, check out some files, and list the entities of the project database.

Note that this example uses aliases for the PCLI commands.

```
# Command Usage: pcli run -sscenario.pcli PDB VLABEL
# Assign the first argument to the PCLI variable PDB
# Assign the second argument to the PCLI variable VLABEL
set -vPDB "$1"
set -vVLABEL "$2"
# $[] implies a run command, so valid run command options
# such as -e can be included.
set -vCURRENT_DIRECTORY $[-e pwd]
echo Creating the project database
cpdb -pr"$CURRENT_DIRECTORY"/Scenario \
    -w"$CURRENT_DIRECTORY"/Scenario/work -nScenario
echo Creating the project
cp -pr"$CURRENT_DIRECTORY"/Scenario \
    -w"$CURRENT_DIRECTORY"/Scenario/work/Project1 Project1
echo Importing archives into Project1
ia -pr"$CURRENT_DIRECTORY"/Scenario -pp/Project1 -c -qw -z \
    "$PDB"/archives/bridge
echo Add version labels to the imported archives from the echo Bridge
    project
run ->foo.tmp LVF -pr"$CURRENT_DIRECTORY"/Scenario \
    -pp/Project1 -aw -z
run -ns -e vcs -v"$VLABEL" '@foo.tmp'
echo Set the work directories for the projects Bridge, res, echo and
    hlp.
swl -pr"$CURRENT_DIRECTORY"/Scenario -pp/Project1/bridge \
    -w"$CURRENT_DIRECTORY"/Scenario/work/
swl -pr"$CURRENT_DIRECTORY"/Scenario -pp/Project1/bridge/hlp \
    -w"$CURRENT_DIRECTORY"/Scenario/work/
swl -pr"$CURRENT_DIRECTORY"/Scenario -pp/Project1/bridge/res \
    -w"$CURRENT_DIRECTORY"/Scenario/work/
echo Check out the files imported to CURRENT_DIRECTORY
run ->bar.tmp LVF -pr"$CURRENT_DIRECTORY"/Scenario \
    -pp/Project1 -aw -z
run -e "get -y -l @bar.tmp"
echo List the entities in the project database
list -pr"$CURRENT_DIRECTORY"/Scenario -aw -l -z /*
echo Cleaning up...
run -e "rm -f foo.tmp"
run -e "rm -f bar.tmp"
echo *******done
```

Example 5: Pass an array to a function.

This example demonstrates how to pass an array variable to a function.

```
array_function()
{
   #Create a local array variable.
   set -l -a -vMYARRAY ""
   #Use Run to evaluate the $1 into the array name and then
   #use Set to copy the array to the local array.
```

```
   Run Set -a$1 -vMYARRAY
}
#Set some array variable.
set -a -vTEST test1 test2 test3
#Call the function with the name of the array variable as
#the argument.
array_function TEST
```

Example 6: Create a simple interactive shell with PCLI.

This example demonstrates how to create an interactive session with the PCLI.

```
# Create an interactive PCLI shell
    # Quit shell by typing "exit"
    While Test 1=1
    {
    # Display prompt
    echo -n "PCLI"
# Read a line of user input
    readline -vCMD
# Execute command. Use Run to
    # evaluate and expand variables
    run -ns $CMD
    }
```

Example 7: Test to see which operating system you are running.

This example demonstrates how to test to determine which operating system you are running. This script uses the UNIX command uname.

```
SetOSDep()
    {
    if test "$ComSpec" != ""
    {
    # Windows
    set -vCMD "$ComSpec /c"
    if test "$TEMP" != ""
            {
            set -vTMPDIR "$TEMP\"
            }
    else
            {
            set -vTMPDIR "C:\TEMP\"
            }
    set -vWHICHOS Win32
    set -vPS "\"
    }
    else
    {
    # UNIX :-)
    set -vCMD ""
    if test "$TMP" != ""
            {
            set -vTMPDIR "$TMP/"
            }
    else
```

```
            {
            set -vTMPDIR "/tmp/"
            }
            set -vPS "/"
            set -vWHICHOS $[-e uname]
            if Test "$WHICHOS" = AIX
            {
            # AIX specific initialization
            }
            elif Test "$WHICHOS" = SunOS
            {
            # SunOS specific initialization
            }
            elif Test "$WHICHOS" = HP-UX
            {
            # HP-UX specific initialization
            }
            elif Test "$WHICHOS" = Linux
            {
            # Linux specific initialization
            }
            else
            {
            echo "UNRECOGNIZED UNIX PLATFORM: $WHICHOS"
            }

    }
    }
```

The above example could be used in conjunction with a script such as the following:

```
### main ###
    SetOSDep
    if Test $WHICHOS = Win32
    {
    set -vDirCMD "dir /b"
    }
    else
    {
    set -vDirCMD "ls"
    }
# Put directory listing in the temporary directory
set -vDirFile "${TMPDIR}dirlist.txt"
run -ns -xo"$DirFile" -e $CMD $DirCMD
# Etc...
```

Chapter 3

PCLI Command Reference

Introduction

This chapter provides a detailed explanation of each PCLI command, and shows how to use the entities and command options appropriate for each.

Each command contains the following sections:

- Description—a description of the command and what action it performs.
- Privileges required—the Version Manager privileges required to execute the command.
- Alias—lists the shortcut alias to the command.
- Exit codes
- Syntax
- Options—the options available for use with the command.
- Examples—ways to use the command with a brief explanation.
- Related Topics—identifies other topics that provide contextual detail for the current topic. This section also provides links between the current topic and other topics to help you navigate through the information in the manual.

If you're using PCLI commands with standard Version Manager commands, see Chapter 1, "Using the Project Command-Line Interface" and the *Command-Line Reference Guide* for more information.

@ command

Use the @ (include) command to insert the contents of a file into a script. This is an easy way to include frequently used functions in scripts, similar to the #include command of the C programming language. For information about using functions see "PCLI Functions" on page 31.

Note, since @ commands are replaced with the contents of the files they point to before the rest of the PCLI commands are run, @ commands:

- ***Cannot*** be made to load conditionally. All @ commands are expanded to reflect the contents of the files they point to, even if the @ commands are located in If command statements that are not run.
- ***Cannot*** be used with a variable in place of an actual path and file name. The @ command must specify the actual path and file name of the file it points to.

Privileges required: None.

Alias: None.

Syntax: **@***file_name*

Or

@/*path/file_name*

Where *file_name* is a file that contains PCLI scripting.

Special Considerations

- All paths and file names specified using the @ command must be valid for the system on which the script is run or the script will terminate. Thus UNIX and Windows systems require separate scripts if paths are specified.
- You can specify a path relative to the location from which you run the script.

Examples

Example 1: Incorporated into a script, the following line would be replaced by the contents of the specified file, which is on a UNIX system.

```
@/usr/serena/include/MyStuff.pcli
```

Example 2: Incorporated into a script, the following line would be replaced by the contents of the specified file, which is on a Windows system.

```
@F:\Serena\vm\include\MyStuff.pcli
```

Example 3: Incorporated into a script, the following line would be replaced by the contents of the specified file, which must be located in the directory from which you invoke the script. That is, if the file `MyStuff.pcli` is in the `/usr/serena/include` directory, you must be at that same directory when you start the script from the command prompt.

```
@MyStuff.pcli
```

AddFiles command

Add individual files or entire directories of files

Use the AddFiles command to add versioned files to a project or folder. You can add a directory tree, a single directory, or individual files to a project database or project. Version Manager creates an archive for each file you add and creates a versioned file that references the new archive.

NOTE On some UNIX systems, the default open file descriptor limit may be set too low. We recommend that you set your file descriptor limit to 128 or higher. For very large databases, we recommend setting the limit as high as allowed by the operating system.

By default, when you add files, the versioned files use the location from which you added the files as the workfile location, unless you specify the **-c** or **-co** options. These options set the workfile location of the files to the workfile location defined for the project or project database to which they are added.

CAUTION! You must specify **-c** or **-co** or the current location of the workfile will be associated with the versioned file. Future check out operations will use this location regardless of the workfile location of the project that contains the file, unless specifically overridden with the **-o** option of the Get command.

When you add versioned files to a project, you must specify a directory or a list of workfiles. You can also use the command to recursively add projects based on the file structure of the directories using the **-z** option. The PCLI creates a project with the same name as each directory, and versioned files are added to the projects. When you recursively add subprojects, the workfile location of the new projects is appended to the workfile location of the parent project.

If you do not enter a *revision* description using the `-m` option, Version Manager uses a revision description of "Initial Revision" when the workfiles are checked in. If you do not

use the -t option, you will be prompted for a description of the versioned file (as opposed to a description of that particular revision of the file). To end the file description, place a period (.) on a line by itself.

NOTE When you add workfiles to a 5.3/6.0 project or folder using the PCLI, you do not have the option of specifying which archive location associated with the project or folder to use as the archive location for the added workfiles. In the Version Manager desktop client, you can select the archive location. The PCLI uses the first archive location specified for the 5.3/6.0 project.

Privileges required

Create Archive and Workfile. For more information on privileges, see the Administrator's Guide.

Alias

AF

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax

AddFiles [*options*] *directory* | *workfile* ...

Where:

directory specifies the directory of files that you want to add. When you add a directory, a project is created and versioned files are added to the project.

workfile specifies the names of the workfiles to add. You can use wildcard characters in file specifications. See the table beginning on page 15 for information about using wildcard characters.

NOTE You can specify more than one directory or workfile, or a combination of the two, in the command line. Separate multiple values with a blank space.

Options

To substitute variables and use list files for additional options, see "Command Options" on page 16 for more information.

-c Copies the workfile to the project workfile location, but does not copy the file if it already exists.

-co Copies the workfile to the project workfile location and overwrites the file if it already exists.

-d Deletes the workfile after check in.

-g **-g** | **-g***promotion_group*

Assigns no promotion group (**-g**) or the specified promotion group (**-g***promotion_group*) to the versioned files as they are added. If a promotion group is specified, it must already exist. If this option is omitted and there is a default promotion group workspace setting defined, then that promotion group is assigned to the versioned files as they are added. See the *User's Guide* for information about default promotion group workspace setting.

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

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[::*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-l Keeps the versioned file locked after it is added to the project and checked in.

-m **-m***description*
-m@*file_name*

Enters a description for the initial revision. If no description is provided, the text "Initial Revision" is used. The *@file_name* option retrieves the description from a file.

-n New archive. If an archive file already exists for the workfile being added, generate one with a unique name.

-ph If an added file is in a subdirectory underneath the project workpath, add the file to a matching subproject. Subprojects are created as necessary.

-pp **-pp***project_path*

Specifies the project or folder to which the files will be added. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Specifies the project database to which the files will be added. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If no project database is specified, the PLCI_PR variable is used. If the variable is not defined, then an error message is displayed and the command is aborted.

-pw **-pw***workpath*

Specifies the project workpath to use instead of the configured project workpath.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

-qw Does **not** add the workfile if a versioned file with the same name already exists and does **not** display a message indicating that the versioned file already exists.

-r **-r***revision*

Specify the initial revision number.

-sp **-sp/@/***userID* | **Public**/*parent_workspace*/*child_workspace*

This option is valid only if you specify either the **-c** or **-co** option. It specifies a public or private workspace. The workfile location of the workspace is used as the workfile location for the files you are adding. Note that user IDs are case-sensitive. To specify the Root Workspace, enter **/@/**RootWorkspace for the workspace value. The specified workspace must already exist. Use the CreateWorkspace command to create a new workspace.

If a workspace is not specified and you have specified **-c** or **-co**, the user's default workspace is used.

-t **-t***description*
-t@*listfile*

Enters a description of the versioned files (as opposed to a description of that particular revision of the files). The *@listfile* option retrieves the description from a file.

NOTE If you do not use the -t option, you will be prompted to enter a description. When prompted, enter the description, using the ENTER key to create new lines as desired. To end the description, place a period (.) on a line by itself.

-v **-v***version_label*

Assigns no version label (**-v**) or the specified version label (**-v***version_label*) to the versioned files as they are added. If this option is omitted and there is a default version defined for the workspace, then that version label is assigned to the versioned files as they are added. See the *User's Guide* for information about default version settings.

Version labels have a limit of 254 characters. You can use alpha, numeric, and special characters except for colons (:), plus signs (+), minus signs (-), and back slashes (\).

-v*version_label*:*

Assigns a floating version label to the versioned files. Floating version labels move with the tip of the trunk or branch to which they are assigned.

-z Includes workfiles in subdirectories and creates projects for all subdirectories. Applies only when you have specified a directory.

Examples **Example 1:** The following example specifies a directory from which to add workfiles; therefore, it creates a project in the project database located in H:\samples and adds all of the workfiles in the directory to the newly created project. The command also uses the **-z** option to create projects for all of the subdirectories located in the \Dev\Work directory and add all of the workfiles in the subdirectories to the newly created projects. A revision

description is also provided (**-m** option), and a promotion group is assigned (**-g** option). The PCLI will prompt for an archive description because an archive description was not specified using the **-t** option.

```
pcli addfiles -prH:\samples -m"first revision" -gDevelopment -z
    F:\dev\work
```

Example 2: The following example adds the workfile Design.java to the Parcheesi project and sets the workfile location for that file to the location set for the annab_ws workspace and copies the workfile from its existing location to the project workfile location (**-c** option). The project database was set using the PCLI_PR variable:

```
pcli addfiles -pp/Parcheesi -sp/@/AnnaB/annab_ws -c Design.java
```

Related Topics

For information about...	See...
Creating projects	"CreateProject command" on page 58
Creating project databases	"CreateProjectDB command" on page 61
Importing Archives	"ImportArchives command" on page 92

AddUser command

Add a user to a project or project database

Use the AddUser command to add a user to a project or project database. The new user will be assigned the Unlimited privilege set.

Privileges required

Unlimited. For more information on privileges, see the Administrator's Guide.

Alias

AU

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax

AddUser [*options*] *name*[:*password*]

Where:

name specifies the ID of the user that you want to add.

password is the password assigned to the user ID, if one is needed.

Options

-e Specifies the expiration date of the user ID.

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

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[::*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp **-pp***project_path*

Specifies the project or folder to which the user will be granted access. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. If neither is specified, the database root is assumed. For more information, see -pp on page 17.

-pr **-pr***project_database*

Specifies the project database to which the user will be granted access. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If no project database is specified, the PLCI_PR variable is used. If the variable is not defined, then an error message is displayed and the command is aborted.

-ps **-ps***privilege*

Specifies one or more privileges to assign to the user. Separate multiple privileges with commas. If the argument contains spaces, enclose it within double quotation marks.

If you do not specify this option, the new user will have the Unlimited privilege set.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

Examples **Example 1:** The following example creates the user Joe in the access control database associated with the project database root assigns him the password Secret. The user

executing this command gains permission to do so by submitting his own password with the -id option.

```
pcli AddUser -prC:\Users\All Users\Serena\VM\SampleDB -idAdmin:PassWord
    Joe:Secret
```

Example 2: The following example creates the user Mary. Her user account will expire on October 31 of 2002.

```
pcli AddUser -pr/export/pdbs/skunkware -e10/31/2002 Mary
```

Related Topics

For information about...	See...
Deleting users	"DeleteUser command" on page 70
Who is accessing specific entities	"WhoAmI command" on page 176

ArraySize command

Use the ArraySize command to return the combined number of elements in the array of one or more variables.

Privileges required: None.

Alias: AS

Exit codes:

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax: **ArraySize** [*options*] *variable_name. . .*

Where *variable_name* specifies the names of the variables for which to return the number of elements in the array. Separate multiple values with a blank space.

Options

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

Example: The following script example returns the number of elements in the array of the variable files.

```
set -a -vfiles test.txt test2.txt test3.txt
```

```
ArraySize files
3
```

Related Topics

For information about...	See...
Other commands used in scripts	Break on page 54, Calc on page 55, Continue on page 57, Echo on page 63, Exit on page 73, For on page 76, If on page 91, Readline on page 135, Return on page 138, Run on page 139, Set on page 142, Test on page 160, and While on page 175.

AssignGroup command

Use the AssignGroup command to assign a specified promotion group to versioned files.

The command may issue a callback asking whether it is okay to replace another revision. One may use the **Run** command with its parameters **-y** (yes to all), **-n** (no to all), and **-np** (do not prompt) to control this callback. The **-q**(quiet) option of the **Run** command can be used to suppress the printing of filenames as they are being processed.

Privileges required: Promotion Group

Alias: AG

Exit codes: Returns no exit code if successful.

-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax: **AssignGroup -g***group* [**-r***revNum*|*label*|*group*] [**-z**] *entity*

Where *group* specifies the name of the promotion group.

Options

-g **-g***group*

Specifies the name of the promotion group to be assigned.

-qr Ignores revisions, labels, and groups that are not located in the archives. Does not print warning messages or set a non-zero exit code.

-r **-r***revNum* | *label* | *group*

Specifies a revision number, label, or group.

-z Recursively assign promotion groups for versioned items in subprojects.

Related Topics

For information about...	See...
Changing to a different promotion group	"ChangeGroup command" on page 56
Deleting promotion groups	"DeleteGroup command" on page 67
Promoting to the next promotion group	"PromoteGroup command" on page 130

Break command

Use the Break command to exit a PCLI For or While loop. Optionally, you can specify a level that instructs the command to break processing at an enclosing loop.

Privileges required None.

Alias None.

Exit codes Returns no exit code if successful.

-2 PCLI command not found
-3 A non-PCLI related error or a command-specific error
-6 An invalid argument was specified
-7 An argument for a flag that is not needed
-8 A missing argument for a flag
-9 Wrong type was specified for an option's argument
-10 The specified file name cannot be read
-11 A required argument is missing
-12 A security exception occurred
-13 An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax **Break** [*options*] [*level*]

Where *level* is an integer that specifies the number of enclosing loops to exit.

Options

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

Example **Example:** The following example breaks from the current and first enclosing loops.

```
break 2
```

Related Topics

For information about...	See...
Other commands used in scripts	ArraySize on page 52, Calc on page 55, Continue on page 57, Echo on page 63, Exit on page 73, For on page 76, If on page 91, Readline on page 135, Return on page 138, Run on page 139, Set on page 142, Test on page 160, and While on page 175.

Calc command

Use Calc evaluate numeric expressions

Calc is a numeric expression evaluator. By default, the results of numeric expressions are printed to standard out. Optionally, you can specify that the results be placed in a variable.

Note that this command supports arbitrary expressions using the operators listed below. It also allows parentheses and nested parentheses to supersede the normal rules of precedence. The normal order of precedence (in decreasing order) is as follows asterisk (*), forward slash (/), percent sign (%), plus sign (+), and minus sign (-).

The following table defines the supported operators.

Command option	Description
A + B	A plus B
A - B	A minus B
A * B	A times B
A / B	A divided by B (integer division)
A % B	A modulo B (integer division remainder)

Privileges required

None.

Alias

None.

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax

`Calc` [*options*] *numeric_expression*

Where *numeric_expression* specifies the expression you want to calculate.

Options

To substitute variables and use list files for additional options, see "Command Options" on page 16 for more information.

-b Specifies that the result of the calculation be expressed as binary.

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

-l Indicates that the variable is a local variable.

-o Specifies that the result of the calculation be expressed as octal.

-v **-v***variable_name*

Specifies the variable into which the calculation will be placed instead of standard out.

-x Specifies that the result of the calculation be expressed as hexadecimal.

Example The following script example uses the Calc command in a While loop to increment the variable i, which is part of the While loop definition. The example goes through a list of directories and adds the files of all of the directories to one Version Manager project. The example first sets a variable array to the directories that contain the files that are to be added and then defines the While loop.

```
set -a -vdirfiles serena/uguide serena/admin serena/readme
set -vi 0
while test $i < $(arraysize dirfiles)
{
   run ->listfiles -e dir $dirfiles[$i]
   addfiles -pr/usrs/docs/vm -pp/release @listfiles
   calc -vi $i+1
}
```

Special Considerations The Calc command reports syntax errors only; such as when illegal characters are used in an expression or if extra operator sequences are included that do not make arithmetic sense. Starting an expression with a minus sign will also result in an error, as the PCLI will think the expression is an option. In this case, use parentheses to surround the expression.

Related Topics

For information about...	See...
Other commands used in scripts	ArraySize on page 52, Break on page 54, Continue on page 57, Echo on page 63, Exit on page 73, For on page 76, If on page 91, Readline on page 135, Return on page 138, Run on page 139, Set on page 142, Test on page 160, and While on page 175.

ChangeGroup command

Use the ChangeGroup command to specify a different promotion group for versioned files that already have a promotion group.

The command may issue a callback asking whether it is okay to replace another revision. One may use the **Run** command with its parameters **-y** (yes to all), **-n** (no to all), and **-np** (do not prompt) to control this callback. The **-q**(quiet) option of the **Run** command can be used to suppress the printing of filenames as they are being processed.

Privileges required: Promotion Group

Alias: CG

Exit codes: Returns no exit code if successful.

-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax: **ChangeGroup -gf***groupFrom* **-gt***rgroupTo* [**-z**] *entity*

Where:

groupFrom specifies the name of the promotion group to change from.

groupTo specifies the name of the promotion group to change to.

Options

-qr Ignores revisions, labels, and groups that are not located in the archives. Does not print warning messages or set a non-zero exit code.

-z Recursively change promotion groups for versioned items in subprojects.

Related Topics

For information about...	See...
Assigning promotion groups	"AssignGroup command" on page 53
Deleting promotion groups	"DeleteGroup command" on page 67
Promoting to the next promotion group	"PromoteGroup command" on page 130

Continue command

Use the Continue command to branch to the top of a For or While loop. Optionally, you can specify a level, which instructs the command to begin processing at the top of an enclosing loop.

Privileges required: None.

Alias None.

Exit codes Returns no exit code if successful.

-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax **Continue** [*options*] [*level*]

Where *level* specifies the number of enclosing loops to continue. If you specify a level that does not exist, an error is generated and processing continues.

Options

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

Example The following example skips commands in the current and first enclosing loops and begins processing in the second enclosing loop.

```
continue 2
```

Related Topics

For information about...	See...
Other commands used in scripts	ArraySize on page 52, Break on page 54, Calc on page 55, Echo on page 63, Exit on page 73, For on page 76, If on page 91, Readline on page 135, Return on page 138, Run on page 139, Set on page 142, Test on page 160, and While on page 175.

CreateProject command

Create projects and 5.3/6.0 folders

Use the CreateProject command to create Version Manager projects or 5.3/6.0 folders.

You can create a new project directly under the project database location or within another project. The parent project in which you are creating the new project must exist.

By default, the workfile location of the new project is set to the workfile location specified in the user's default workspace. Optionally, you can use the **-sp** option to set a different workspace from which to take the workfile location. Or, you can use the **-w** option to set the workfile location manually.

For Version Manager 5.3/6.0 project roots, this command creates both projects and folders. These project roots contain one level of projects and a level of folders. Depending on the location specified when you use the command, either a project or folder is created.

NOTE To make the new project visible in the desktop client, select the project database and then select View | Refresh.

Privileges required: Create Project. For more information on privileges, see the *Administrator's Guide.*

Alias: CP

Exit codes:

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax: **CreateProject** [*options*] *project_name*

Where *project_name* specifies the name of the new project, and, optionally, its parent path. You need only specify the parent path if you do not set the current project with the **-pp** option or the PCLI_PP variable. Note that the project name cannot begin or end with a tab or blank space. Any character can be used in the name except an asterisk (*), a colon (:), a vertical bar (|), forward and backward slashes (/ \), a question mark (?), and angle brackets (< >). No project name can be the two character name of .. or the one character name of . or @.

Options

To substitute variables and use list files for additional options, see "Command Options" on page 16 for more information.

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

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.

For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp **-pp***project_path*

Specifies the project in which to create the project or folder. You do not need to specify this option, which sets the current project, if you are creating a project directly beneath the project database or if you specify the parent project of the new project in the *project_name* argument. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Specifies the location of the project database in which you are creating the project. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If no project database is specified, the PLCI_PR variable is used. If it is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

-sp **-sp/@/***userID* | **Public**/*parent_workspace*/*child_workspace*

Specifies a public or private workspace. The workfile location of the workspace is used as the workfile location for the project you are creating. Note that user IDs are case-sensitive. To specify the Root Workspace, enter **/@/**RootWorkspace for the workspace value.

The specified workspace must already exist. Use the CreateWorkspace command to create a new workspace.

-w **-w***workfile_location*

Defines the workfile location of the project if you want a location other than the location in the workspace specified with the **-sp** option or the location in the user's default workspace if the **-sp** option is not specified.

Examples **Example 1:** The following examples create a project and subproject. The first line creates the VolleyBall project in a 5.3/6.0 project root. The second command creates a subproject of Setters in the VolleyBall project. The workfile location for both of these projects is set to the workfile location specified in the user's default workspace because no other workfile location or workspace is specified.

```
pcli CreateProject -prH:\VM60Games VolleyBall
pcli CreateProject -prH:\VM60Games -pp/VolleyBall Setters
```

Example 2: The following example creates a project named Parcheesi in the H:\BoardGames project database and specifies a workfile location on a common server to which all users have access:

```
pcli CreateProject -prH:\BoardGames -wS:\BoardGames\parcheesi\work -pp/ Parcheesi
```

Related Topics

For information about...	See...
Creating project databases	"CreateProjectDB command" on page 61
Adding workfiles	"AddFiles command" on page 46
Importing Archives	"ImportArchives command" on page 92

CreateProjectDB command

Create project databases or project roots

Use the CreateProjectDB command to create Version Manager project databases and Version Manager 5.3/6.0 project roots.

Note that you cannot create a project database under another project database, nor should you create a project database in the same location as a 5.3/6.0 project root.

The directories and files that are created, by default, when you create a project database are as follows:

- *Archives* directory, which is set as the default archive directory for the project database. To change the archive location associated with a project database, use the SetArchiveLocation command.
- A configuration file is placed in the archives directory. For security purposes, Version Manager masks the name of this configuration file. For example, this file may be assigned a name such as "c6bonpj1.cfg." To change the configuration file associated with a project database, use the SetConfigFile command.

NOTE For non-file-server project databases, the login source is set to HOST. For file-server project databases, the login source is set to VLOGIN.

- An access control database is placed in the archives directory. For non-file-server project databases, this file is not enabled in the configuration file. For file-server project databases, the access control database is enabled. For security purposes, Version Manager masks the name of this access control database. For example, this file may be assigned a name such as "c6bonpj1.db." For more information on access control databases, see the Administrator's Guide.
- *Pvcsuser* directory, which contains user information such as private workspace settings.
- A *Lib* directory, which contains information about the Version Manager release you used to create the project database.
- A tools configuration file.
- A system identification file and a metadata file.

Privileges required: None.

Alias: `CPDB`

Exit codes:

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error

-6 An invalid argument was specified
-7 An argument for a flag that is not needed
-8 A missing argument for a flag
-9 Wrong type was specified for an option's argument
-10 The specified file name cannot be read
-11 A required argument is missing
-12 A security exception occurred
-13 An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax **CreateProjectDB -pr***database_location* [**-n***projectdb_name*]
-w*workfile_location* [*options*]

Where:

database_location specifies the location of the new project database. The **-pr** option is required.

projectdb_name specifies the name of the new project database. This applies only to project databases. The **-n** option is required when you are creating a project database.

workfile_location specifies the workfile location for the new project database. The **-w** option is required.

Options

To substitute variables and use list files for options, see "Command Options" on page 16 for more information.

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

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

Specifies a user ID and password, if necessary. This option is used only for creating project databases that have a VLOGIN (Login Dialog) login source specified and an access control database enabled. This is the default state for a project database created on a Version Manager file server.

-n **-n***projectdb_name*

Specifies the name of the new project database. This applies only to project databases. This is required for creating a project database.

-pa **-pa***password*

Specifies the password required to create a project database **if** your Administrator has restricted the creation of project databases.

-pr **-pr***database_location*

Specifies the location of the new project database. This is required.

-t **-t**{[file]|[file53]}

Specifies the type of project database to create. The default project database type is `file`, which means to create a project database. Specifying `file53` creates a Version Manager 5.3/6.0 project root.

-w **-w***workfile_location*

Specifies the workfile location for the new project database. This is required.

Examples **Example 1:** The following example creates a project database named BoardGames and specifies a workfile location on a local drive.

```
pcli createprojectdb -prH:\BoardGames -nBoardGames -wC:\BoardGames\work
```

Example 2: The following example creates a 5.3/6.0 project root in a directory named ServerPrjs:

```
pcli createprojectdb -prH:\ServerPrjs -wH:\ServerPrjs\Work -tfile53
```

Related Topics

For information about...	See...
Creating projects	"CreateProject command" on page 58
Adding workfiles	"AddFiles command" on page 46
Importing Archives	"ImportArchives command" on page 92

CreateWorkspace command

Use the CreateWorkspace command to create a new private or public workspace. A workspace is a collection of work settings defined for a project database, which includes the work settings for all of the projects and versioned files contained within the project database. These settings are workfile location, default version, base version, and branch version.

When you create a workspace, you must specify the workspace from which the new workspace will inherit its settings; the workspace you specify is the parent workspace. The parent of a public workspace must be a public workspace. The parent of a private workspace can be either a public or private workspace.

See the *User's Guide* for complete information about workspaces.

Privileges required None.

Alias CW

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax **CreateWorkspace** [*options*] *workspace*

Where *workspace* specifies the workspace path of the workspace to create. To specify a private workspace, you use the following syntax:
/@/*userID*/*parent_workspace/*[*child_workspace*]

where *userID* is your user ID and is case sensitive. For example, /@/AdamJ/myprivateworkspaces/mylocaldrive. In this case, myprivateworkspaces is the parent workspace for mylocaldrive.

To specify a public workspace, you use the following syntax:
/@/Public/*parent_workspace/*[*child_workspace*]

For example, /@/Public/networkdrive. In this example, there is no child workspace specified. In this case, the parent workspace for networkdrive is the Root Workspace.

Options

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

-pr Specifies the project database for which to create a workspace. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If no project database is specified, the value of the PLCI_PR variable is used. If it is not defined, then an error message is displayed and the command is aborted.

-pw Specifies the public parent of a private workspace. The full workspace path of the public workspace must be specified. See the information above in the Syntax section for how to specify workspace paths. You need only use this option when you are creating a private workspace that will have a public workspace as its parent. Otherwise, the parent workspace is specified in the workspace path.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

Examples **Example 1:** The following example creates a public workspace named localserver. Its parent is the Root Workspace.

```
pcli createworkspace -prD:\sample /@/Public/localserver
```

Example 2: The following example creates a public workspace named vol1. Its parent is localserver.

```
pcli createworkspace -prD:\sample /@/Public/localserver/vol1
```

Example 3: The following example creates a private workspace named proj1. Its parent is a private workspace named mywork. Then, the example sets the workfile location of this new private workspace for the Proj1 project of the project database located in D:\sample.

```
pcli createworkspace -prD:\sample /@/mikaylap/mywork/proj1
pcli setworklocation -prD:\sample -pp/Proj1 -sp/@/mikaylap/mywork/proj1 -wD:\mywork\proj1
```

Example 4: The following example creates a private workspace that has a public workspace as its parent. The **-pw** option specifies the public workspace as the parent.

```
pcli createworkspace -prD:\sample -pw/@/Public/localserver /@/adamj/adamj_proj1
```

Related Topics

For information about...	See...
Setting workfile locations	"SetWorkLocation command" on page 155

Delete command

Use the Delete command to delete folders, projects, versioned items, and workspaces.

Privileges required

Depending on the type of entity you are operating on, you must have one of the following privileges.

Entity Type	Privilege Required
Versioned files and folders	Add or Remove Versioned Files
Projects	Delete Project
Workspaces	Unlimited

For more information on privileges, see the Administrator's Guide.

Alias

None.

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax

Delete [*options*] *entity ...*

Where *entity* specifies the item that you want to delete.

Options

-h

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

-id

-id*user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp **-pp***project_path*

Sets the current project or folder for this command execution. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Sets the current project database for this command execution. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If no project database is specified, the PLCI_PR variable is used. If the variable is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

Examples **Example 1:** The following example shows two ways to delete the versioned files `bridge.h` and `Readme.txt` from the project /bridge. As with the desktop client, deleting versioned files does not remove the corresponding archives.

```
pcli Delete -prC:\Users\All Users\Serena\VM -idAdmin /bridge/bridge.h /
    bridge/Readme.txt
pcli Delete -prC:\Users\All Users\Serena\VM\SampleDb -idAdmin -pp/
    bridge bridge.h Readme.txt
```

Example 2: The following example deletes the project /chess and all of its subprojects. Note that recursive behavior is implied when you are operating on a project.

```
pcli Delete -prC:\Users\All Users\Serena\VM\SampleDb -idAdmin /chess
```

Example 3: The following example deletes the public workspace QA.

```
pcli Delete -prC:\Users\All Users\Serena\VM\SampleDb -idAdmin /@/
    Public/QA
```

Example 4: The following example deletes the private workspace MySpace for the user Joe.

```
pcli Delete -prC:\Users\All Users\Serena\VM\SampleDb -idJoe /@/Joe/
    MySpace
```

Related Topics

For information about...	See...
Deleting users	"DeleteUser command" on page 70
Where archives are located	"GetArchiveLocation command" on page 84
Listing entity information	"List command" on page 102
Listing the versioned files in a project	"ListVersionedFiles command" on page 122

DeleteGroup command

Use the DeleteGroup command to remove a specified promotion group from versioned files.

Privileges required: Promotion Group

Alias: DG

Exit codes: Returns no exit code if successful.

-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax: **DeleteGroup -g***group* [-z] *entity*

Where *group* specifies the name of the promotion group.

Options

-qr Ignores revisions, labels, and groups that are not located in the archives. Does not print warning messages or set a non-zero exit code.

-z Recursively remove promotion groups for versioned items in subprojects.

Related Topics

For information about...	See...
Assigning promotion groups	"AssignGroup command" on page 53
Changing to a different promotion group	"ChangeGroup command" on page 56
Promoting to the next promotion group	"PromoteGroup command" on page 130

DeleteLabel command

Use the DeleteLabel command to remove a specified version label from specified versioned files or projects.

Privileges required: Delete Version Label. For more information on privileges, see the Administrator's Guide.

Alias: DL

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax: **DeleteLabel** [*options*] **-v***version entity ...*

Where *entity* specifies the versioned file, project, or project database from which you want to remove a version label and *version* specifies the version label.

Options

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

-id: **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id**:*PassWord*.

-idCAC_LOGIN: **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp **-pp***project_path*

Sets the current project or folder for this command execution. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Sets the current project database for this command execution. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If no project database is specified, the PLCI_PR variable is used. If the variable is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qr Ignores revisions, labels, and groups that are not located in the archives. Does not print warning messages or set a non-zero exit code.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

-sp **-sp**{[/@/*userID*] | [Public/*parent_workspace*/*child_workspace*]}

Specifies a public or private workspace. Note that user IDs are case-sensitive. To specify the Root Workspace, enter /@/RootWorkspace for the workspace value. If a workspace is not specified, the user's default workspace is used.

-v **-v***version*

Specifies the version label to be removed. This is required.

-z Removes a label from all versioned files in the specified project and its subprojects.

Examples **Example 1:** The following example shows two ways to remove the label Test Label from the versioned files /bridge/bridge.h and /bridge/ReadMe.txt.

```
pcli DeleteLabel -prC:\Users\All Users\Serena\VM\SampleDb -idAdmin
   -v"Test Label" /bridge/bridge.h /bridge/ReadMe.txt
pcli DeleteLabel -prC:\Users\All Users\Serena\VM\SampleDb -idAdmin
   -v"Test Label" -pp/bridge bridge.h ReadMe.txt
```

Example 2: The following example removes the label Release1.23 from all versioned files in project /bridge, but not from its subprojects.

```
pcli DeleteLabel -prC:\Users\All Users\Serena\VM\SampleDb -idAdmin
   -vRelease1.23 /bridge
```

Example 3: The following example removes the label Release1.23 from all versioned files in project /bridge, including its subprojects.

```
pcli DeleteLabel -prC:\Users\All Users\Serena\VM\SampleDb -idAdmin
```

```
-vRelease1.23 -z /bridge
```

Related Topics

For information about...	See...
Adding version labels	"Label command" on page 100
Listing floating labels	"ListFloatingLabels command" on page 114

DeleteUser command

Use the DeleteUser command to delete users from the access control database. This also deletes the user and his workspaces from the `pvcsuser` directory of the project database.

Privileges required: Unlimited. For more information on privileges, see the Administrator's Guide.

Alias: `DU`

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax: **`Delete`** [*`options`*] *`user_ID ...`*

Where *`user_ID`* specifies the user that you want to delete.

Options

-h — Displays help for the command. The action terminates after it processes the **`-h`** option even if you specify other options.

-id — **`-id`***`user id`*[`:`*`user_password`*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **`-id`** to specify your user ID—and password too if a password is assigned to your user ID. For example, **`-id`***`UserID`*`:`*`Password`*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **`-id`** to enter your password, if you have one. For example, **`-id:`***`PassWord`*.

-idCAC_LOGIN — **`-id`**`CAC_LOGIN[:`*`PIN`*`:`*`Aliasname`*`]`

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp **-pp***project_path*

Sets the current project or folder for this command execution. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Sets the current project database for this command execution. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If no project database is specified, the PLCI_PR variable is used. If the variable is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

Examples **Example 1:** In the following example the user Admin deletes the user Joe from the project database.

```
pcli DeleteUser -prC:\Users\All Users\Serena\VM\SampleDb -idAdmin Joe
```

Related Topics

For information about...	See...
Adding users	"AddUser command" on page 50
Who is accessing specific entities	"WhoAmI command" on page 176

Echo command

List contents of variables Use the Echo command to output literal text or the contents of PCLI variables.

Privileges required None.

Alias None.

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument

-10 The specified file name cannot be read
-11 A required argument is missing
-12 A security exception occurred
-13 An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax **echo** [*options*] ${*variable_name*} | *text*

Where:

variable_name specifies a PCLI variable. The PCLI displays the contents of the variable.

text specifies the text to display.

Options

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

-n Omits any newlines that are contained within the text to be displayed. This option enables you to control the format of the output.

-ns Prevents quotation marks from being stripped from all arguments except those that are listed as options of the Echo command itself.

-r Converts the following backslash escape combinations within the text to be displayed into their unprintable counterparts:

\n = newline
\r = return
\t = tab
\f = form feed
\b = backspace

When this option is used, outputting a backslash requires double backslashes.

-tq Trims the quotes from the start and end of an argument. Use this option to expand variables from double quoted strings and preserve any double quotes in the string's value. For example:

```
set VAR='This variable has "double quotes" in its value'
set -tq VAR_COPY="${VAR}"
echo -tq "VAR_COPY is set to: ${VAR_COPY}."
```

This example prints:

```
VAR_COPY is set to: This variable has "double quotes" in its value.
```

Examples **Example 1:** The following example returns the value of the PCLI_PR variable set using the Set command:

```
echo ${PCLI_PR}
H:\BoardGames
```

Example 2: The following script uses the List command and places the output in a temporary file, and then sets the variable to the value of the file's contents. The script then uses the Echo command to return the contents of the variable:

```
# Using the List command and placing the output in a file.
run ->list.tmp list -prD:\producta -caWorkPath /Proj5
# Setting variable to the value of the contents of list.tmp.
Set -vfiles @list.tmp
Echo The following was generated from the list command.
# Echoing the contents of the variable.
Echo ${files}
```

Related Topics

For information about...	See...
Setting variables	"Set command" on page 142
Other commands used in scripts	ArraySize on page 52, Break on page 54, Calc on page 55, Continue on page 57, Exit on page 73, For on page 76, If on page 91, Readline on page 135, Return on page 138, Run on page 139, Test on page 160, and While on page 175.

Exit command

Use the Exit command to stop PCLI processing and return the specified status to the calling program.

Privileges required: None.

Alias: None.

Exit codes: Returns no exit code if successful.

-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax: **Exit** [*options*] [*status*]

Where *status* specifies the status to return. If you do not specify *status*, a 0 (zero) status is returned.

Options

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

Example The following examples demonstrates the use of the Exit command in an If statement.

```
set -vConfigfile $(getconfigfile -prD:\producta)
If Test $Configfile=H:\producta\archives\cb09zjk.cfg
{
   exit
}
else
{
   setconfigfile -prD:\producta -cH:\producta\archives\cb09zjk.cfg
}
```

Related Topics

For information about...	See...
Other commands used in scripts	ArraySize on page 52, Break on page 54, Calc on page 55, Continue on page 57, Echo on page 63, For on page 76, If on page 91, Readline on page 135, Return on page 138, Run on page 139, Set on page 142, Test on page 160, and While on page 175.

ExportPDB command

Send output of project database to an file

Use the ExportPDB command to export project database and project information into an INI file format. Workfiles, configuration files, access control database files, and archives are not copied, but the command exports the references to these files that are stored in the project database.

If an output file is not specified, then output is printed to standard out. If projects are not specified, the entire project database is exported, meaning all of the projects within the project database. Otherwise, only the specified projects of a project database are exported.

The exported project database information can then be used by the ImportPDB command to create a new project database. If you specify to export a subproject of a project and not to export the parent project of the subproject, the subproject cannot be imported using the ImportPDB command because the PCLI does not have the information about the parent project.

NOTE You can use this command to export project database information of Version Manager 5.3/6.0 project roots **only if** the project roots have been operated on using the current edition of the Version Manager desktop client. In this case, only the project database type (5.3/6.0), workspaces, and users are exported.

The following is an example of the INI file format of a project database.

```
[PDB]
```

```
Version=1.0
RootPath=D:\pdb2
RootName=Producta
RootType=file
[/]
Type=ProjectRoot
WorkPath=D:\pdb2\work
ArchiveLocation=D:\pdb2\arachives
WorkPathNamespace=0
VersionSelection=0
ConfigFlags=3
ConfigPath=D:\pdb2\archivesct4fv0d1.cfg
Name=Producta
[/Proj1]
Type=Project
WorkPath=Proj1
ArchiveLocation=Proj1
WorkPathNamespace=0
VersionSelection=0
ConfigFlags=0
Name=Proj1
```

Privileges required: SuperUser. For more information on privileges, see the Administrator's Guide.

Alias: `EPDB`

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax: **`ExportPDB`** [*`options`*] [*`entity...`*]

Where *`entity`* is a project. You need specify a project only if it is different from the current project. Specify a project only if you want to export a specific project from the project database and not all of the projects contained in the project database. You can specify more than one project on the command line. Separate multiple values with a blank space.

NOTE The current project is set using either the **`-pp`** option (as described below) or the PCLI_PP variable.

Options

To substitute variables and use list files for additional options, see "Command Options" on page 16 for more information.

-f **-f***output_file*

Names the file that will contain the exported information. Otherwise, the output is printed to standard out.

-h Display help for the command. The action terminates after it processes the **-h** option even if you specify other options.

-pp **-pp***project_path*

Specifies the project for which to export information. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Specifies which project database to export. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If no project database is specified, the value of the PLCI_PR variable is used. If it is not defined, then an error message is displayed and the command is aborted.

-z Recursively exports all subprojects.

Examples **Example 1:** The following example exports all of the projects and subprojects in the project database located in H:\Samples.

```
pcli exportpdb -prH:\Samples -z -fsampleout.txt
```

Example 2: The following example exports only the Prj1 project in the project database located in H:\Samples. Note that the PCLI_PR variable was set to the location of the project database, and therefore did not need to be specified:

```
pcli exportpdb -fproj1out.txt /prj1
```

Related Topics

For information about...	See...
Importing project databases	"ImportPDB command" on page 96
Listing project databases	"ListProjectDB command" on page 116

For command

Use the For command to loop through all of the items in a variable's array or loop through the value of each variable specified.

This command implements a typical scripting For. . .In loop. For each iteration of the loop, the specified local variable is set to each option value one after the other.

Privileges required None.

Alias None.

Exit codes Returns the value of the last command executed in the For loop when successful.
-2 PCLI command not found
-3 A non-PCLI related error or a command-specific error
-6 An invalid argument was specified

-7 An argument for a flag that is not needed
-8 A missing argument for a flag
-9 Wrong type was specified for an option's argument
-10 The specified file name cannot be read
-11 A required argument is missing
-12 A security exception occurred
-13 An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax

```
For [options] local_variable in variable_name...
 {
   action
     ...
 }
```

Where:

local_variable specifies the local variable to set in each iteration of the For loop.

variable_name specifies the variable from which to take the value and set it to the local variable. You can specify multiple variables. Separate each value with a blank space.

action is the action to take in the loop. This action can be a PCLI command or any external program.

Options

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

Example **Example 1:** The following example uses two For loops. The first For loop creates the `-c` option for the Get and Put commands. The `-c` option specifies the configuration files that the Get and Put commands use when you check in and out files. The second For loop goes through a list of versioned files. The action taken on the files is to check out the files, add a copyright to the beginning of each file, and then check the files back in. The external commands cp, cat, and rm, are UNIX commands.

```
#Get the configuration files used by proj1.
#The output contains only the paths to the config files.
run -vaconfigfiles getconfigfile -pr/users/producta -a /proj1
#Create the -c option for the Get and Put commands by
#prepending a -c to each configuration file path and storing
#it in the configargs variable.
for i in $configfiles[]
{
   set -ns -vconfigargs ${configargs} "-c$i"
}
#Create a list of versioned files with the format
#workfile(archive) to be used by the Get command.
run ->listfiles listversionedfiles -pr/users/producta -aw /proj1
run -ns -e get -l $configargs '@listfiles'
#Create a For loop to loop through each workfile file
#location and add the copyright info to the beginning of the file.
for i in $[listversionedfiles -w -pr/users/producta /proj1]
{
   run -ns -e cp "${i}" tmpfile
```

```
   run -ns -e cat copyright tmpfile > "${i}"
   run -e rm tmpfile
}
#Check in the updated files.
run -ns -e put $configargs '@listfiles'
run -e rm listfiles
```

Example 2: The following example shows you how to use:

- For loops when arguments are passed with trailing and/or leading spaces
- The variable PCLI_TRIM_AUTOVAR to control the behavior of For loops.

Sometimes trailing and/or leading spaces are required to access data and to pass arguments correctly.

Let's say we have TestAutoVar.pcli PCLI script file:

```
GetData()
{
      echo " LeadingSpace"
      echo "Middle Space"
      echo "TrailingSpace "
}

RunTest()
{
      for arg in $[GetData]
      {
            echo -ns arg="$arg"
      }
}
RunTest

echo Enabling PCLI_TRIM_AUTOVAR
set PCLI_TRIM_AUTOVAR=true

RunTest
```

And then script file TestAutoVar.pcli is executed with command:

```
pcli run -sTestAutoVar.pcli
```

```
arg=" LeadingSpace"
arg="Middle Space"
arg="TrailingSpace "
Enabling PCLI_TRIM_AUTOVAR
arg="LeadingSpace"
arg="Middle Space"
arg="TrailingSpace"
```

Related Topics

For information about...	See...
Setting variables	"Set command" on page 142
Other commands used in scripts	ArraySize on page 52, Break on page 54, Calc on page 55, Continue on page 57, Echo on page 63, Exit on page 73, If on page 91, Readline on page 135, Return on page 138, Run on page 139, Test on page 160, and While on page 175.

Get command

Use the Get command to check out the specified versioned files or the versioned files contained within specified projects.

Privileges required

Depending on the type of revision you are operating on and whether you are locking it, you must have the following privileges.

Operation Type	Privileges Required
Get tip	Check Out Tip
Get and lock tip	Check Out Tip Lock Tip
Get non-tip	Check Out Non-Tip
Get and lock non-tip	Check Out Non-Tip Lock Non-Tip

For more information on privileges, see the Administrator's Guide.

Alias

None.

Exit codes

0 Successful command completion
-2 PCLI command not found
-3 A non-PCLI related error or a command-specific error
-6 An invalid argument was specified
-7 An argument for a flag that is not needed
-8 A missing argument for a flag
-9 Wrong type was specified for an option's argument
-10 The specified file name cannot be read
-11 A required argument is missing
-12 A security exception occurred
-13 An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax

Get [*options*] *entity* ...

Where entity specifies a versioned file, folder, project, or project database from which you want to check out a revision

Options

-a **-a***path*

Specifies an alternate location to place workfiles, rather than the location defined in the workspace. See also the **-o** and **-bp** options.

For single file checkouts, the alternate location can be either a directory or an alternate name for the file itself. If the specified leafname of the destination path:

- Does not exist and the **-bp** option was **not** used, the leafname is assumed to be an alternate filename.
- Already exists as a directory, the file will be placed into that directory using its original name.

If the **-bp** option is used, the leafname is always assumed to be a directory; additional subdirectories may be created depending on how the **-bp** option was used.

-bp **-bp***path*

Specifies the base project path to use in calculating workfile locations when the **-a** option has been specified. When you operate on multiple files, *Path* must be the entity path to a common parent of each of the items being checked out. If this option is not specified, the command uses the first parent that is common to all the entities you are operating on.

-d **-d***date/time*

Specifies a revision by the date and time it was checked in. The command will operate on the last revision that was checked in before the specified date and time. If you specify only a date, the time defaults to midnight. For more information on specifying dates and times, see "Date and Time Formats" on page 30.

-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***group*

Specifies a promotion group. This option is required if the -fg parameter is set. This option functions in two different ways, depending on whether or not the revision is being locked with the **-l** option:

- If the revision is not being locked, **-g** specifies the lowest level promotion group in which to look for a revision. The search continues up the hierarchy of promotion groups until the revision is found. If the revision is not found, the get operation fails.
- If the revision is being locked (**-l**), **-g** specifies the lowest level promotion group to which to assign the locked revision. If this option is omitted, the default promotion group is assigned to the checked out revision, if one exists.

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

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-l **-l**[*revision*]

Locks the revision of the file you are getting. Optionally, allows you to specify the revision to lock. By default, the default revision defined for the workspace is acted on. Note you can also specify the revision with the **-r** option.

-nb No branching. This option prevents you from locking a revision if a lock would result in a branch upon check in. This option is ignored if the BranchWarn directive is not in effect.

-nm No multilock. This option overrides the MultiLock directive and prevents you from applying another lock to a revision that is already locked.

-o Overrides the workfile locations defined in the project and versioned files, and instead uses a hierarchy of directories that mirror the structure and names of the project and subprojects.

Note if you do not use this option, any versioned file or project that has an absolute workfile location associated with it will be copied to that workfile location, even if you specify a workspace or use the **-a** option.

-p Pipes the revision to stdout.

-pp **-pp***project_path*

Sets the current project for this command execution. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Sets the current project database for this command execution. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If it is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qr Ignores revisions, labels, and groups that are not located in the archives. Does not print warning messages or set a non-zero exit code.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

-r **-r***revision*

Specifies the revision, promotion group, or version to act upon.

Note the operation will fail if the exact promotion group does not exist in the versioned file. Use the -g option to continue searching for a match higher in the promotion model hierarchy.

-sp **-sp**{[/@/*userID*] | [Public/*parent_workspace*/*child_workspace*]

Specifies a public or private workspace. Note that user IDs are case-sensitive. To specify the Root Workspace, enter /@/RootWorkspace for the workspace value. If a workspace is not specified, the user's default workspace is used.

-t **-t**[*revision*]

Touch. Sets the workfile to the current date and time after getting the revision. Optionally, allows you to specify the revision to obtain. By default, the default revision associated with the workspace is acted on. Note you can also specify the revision with the -r option.

-u **-u**[*date/time*]

Gets the revision only if it is newer than the current workfile or the date/time specified. For more information on specifying dates and times, see "Date and Time Formats" on page 30.

-v **-v***version*

Gets the revision associated with the specified version label.

-w **-w**[*revision*]

Makes the workfile writable without locking the revision. Optionally, allows you to specify the revision to obtain. By default, the default revision associated with the workspace is acted on. Note you can also specify the revision with the **-r** option.

-yb Yes to branching. Overrides the BranchWarn directive to allow you to lock a revision even if that will result in a branch upon check in. This option works in conjunction with the **-l** option.

-ym Yes to multilock. If the MultiLock directive is in effect, this option will apply another lock without first prompting for confirmation.

-z Includes versioned files in subprojects.

Examples **Example 1:** The following example recursively gets all revisions in the Project1 project database that are associated with the version label Beta2 and copies the workfiles to the C:\John\Project1 directory based on the project structure.

```
pcli Get -prD:\PDBs\Project1 -aC:\John\Project1 -o -vBeta2 -z /
```

Example 2: The following example gets the revisions in the Project1 project database that are located in or below the /bugfixes and /windows/dialogs projects and are associated with the QA promotion group or higher. These workfiles are then copied to the default workfile location as defined in the default workspace for the active user.

```
pcli Get -prD:\PDBs\Project1 -gQA /bugfixes /windows/dialogs
```

Example 3: The following example shows two ways to get the revisions in the Project1 project database that are located in or below the /src/binaries and /src/classes projects and copy the workfiles to the C:\builds directory based on the project structure. Since the -bp option is not specified, the first common parent (/src) is assumed. As a result, the target directory, C:\builds, will contain the subdirectories binaries and classes.

```
pcli Get -prD:\PDBs\Project1 -aC:\builds -o /src/binaries /src/classes
pcli Get -prD:\PDBs\Project1 -aC:\builds -o -pp/src binaries classes
```

Example 4: This is like the previous example, but uses the -bp option to specify that the base path is / (the project database root). As a result, the target directory, C:\builds, will contain the subdirectory src, which will contain the subdirectories binaries and classes.

```
pcli Get -prD:\PDBs\Project1 -aC:\builds -o -bp/ /src/binaries /src/classes
pcli Get -prD:\PDBs\Project1 -aC:\builds -o -bp/ -pp/src binaries classes
```

Example 5: The following example shows how to use the Run command in conjunction with get to pass a Yes or No answer to all prompts. Note, if any of your arguments are quoted, use the -ns option to prevent stripping of the quotation marks. This example assumes that the project database is defined in PCLI_PR variable.

```
pcli Run -y -ns Get "/My Files/foobar.txt"
pcli Run -n -ns Get "/My Files/foobar.txt"
```

Example 6: The following examples illustrate how the behavior of the -a option differs depending on how it is used and what it is used on.

6a (-a without -bp):

```
pcli get -pr"C:\Users\All Users\Serena\VM\SampleDB" -idAdmin
    -aC:\Alternate\Path /chess/server/server.bat
```

If the path C:\Alternate\Path does not exist, then the target becomes a file named Path:

```
C:\Alternate\Path <- /chess/server/server.bat Checked out revision 1.0.
```

If the path C:\Alternate\Path exists as a directory, then the file is checked out into this directory using the file's original name:

```
C:\Alternate\Path\server.bat <- /chess/server/server.bat Checked out
    revision 1.0.
```

6b (-a with -bp):

```
pcli get -pr"C:\Users\All Users\Serena\VM\SampleDB" -idAdmin
    -aC:\Alternate\Path -bp/chess/server /chess/server/server.bat
```

In this case, the path C:\Alternate\Path did not exist, but the -bp option specified that it should be a directory at the hierarchical level of /chess/server. The path was thus created as a directory and the file was checked out into this directory using the file's original name:

```
C:\Alternate\Path\server.bat <- /chess/server/server.bat Checked out
    revision 1.0.
```

Related Topics

For information about...	See...
Listing the workfile location of projects and versioned files	"GetWorkLocation command" on page 88
Locking revisions	"Lock command" on page 126
Checking in revisions	"Put command" on page 131
Using the Run command	"Run command" on page 139

GetArchiveLocation command

List archive locations for the specified entity

Use the GetArchiveLocation command to list the archive location for the specified project database, project, or versioned file.

When you specify a project database or versioned file, the command always returns the full path of the archive location.

For projects, you can choose to return either the full path to the archive location (this is the default if the -a flag is not specified) or only the portion of the path that is stored in the project. This partial path is relative to the project's parent archive location. For more information on how Version Manager stores relative paths, see the *User's Guide.*

NOTE This command is not functional for Version Manager 5.3/6.0 project roots.

Privileges required

None.

Alias

GAL

Exit codes

0	Successful command completion
1	Improper usage of command
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax

GetArchiveLocation [*options*] [*entity*]

Where *entity* specifies a project or versioned file. Versioned file names are case-sensitive. You need not specify an entity if you want to get the archive location for the current project or project database.

NOTE The current project is set using either the **-pp** option (as described below) or the PCLI_PP variable. A project database is set by using either the **-pr** option (as described below) or the PCLI_PR variable.

Options

To substitute variables and use list files for additional options, see "Command Options" on page 16 for more information.

-a Lists only the portion of the archive location stored within the selected project. This option allows the user to see the string that was set using the SetArchiveLocation command and to determine if the archive location is relative to the parent or is absolute.

This option only affects projects, because the archive location for versioned files is always a full path.

-h Display help for the command. The action terminates after it processes the **-h** option even if you specify other options.

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp **-pp***project_path*

Sets the current project for this command execution. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Sets the current project database for this command execution. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If it is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

Examples **Example 1:** The following example returns the archive location for the Dialogs project in the Online Help project database. Note that the path to the project database is quoted because the directory names contain spaces. Also, an entity (project, in this case) is specified because no current project is set.

```
pcli getarchivelocation -pr"Y:\doc\prjs\Online help" /content/dialogs
    Y:\doc\prjs\Online help\archives\Content\Dialogs
```

You can return the same result by substituting the -pp option for the /content/dialogs entity in the previous example:

```
pcli getarchivelocation -pr"Y:\doc\prjs\Online help" -pp/content/dialogs
```

Example 2: The following example returns the archive location for a versioned file. The project database location was set using the PCLI_PR environment variable. Note that the name of the versioned file is case-sensitive.

```
pcli getarchivelocation /content/dialogs/ACCESSLIST.HTML
   Y:\dpc\prjs\Online help\arachives\Content\Dialogs\ACCESSLIST.HTML-arc
```

Example 3: The following two commands return first the full path of the archive location (this is the default). The second command returns only the portion of the archive location that is different from the project's parent archive location. Note that the command line uses the alias for the command, gal.

```
pcli gal -prH:\Games -pp/VolleyBall/Setters
H:\Games\archives\VolleyBall\Setters
pcli gal -prH:\Games -pp/VolleyBall/Setters -a
Setters
```

Example 4: The following example returns the archive location for a project within a project database that requires a user id and password.

```
pcli gal -prD:\productb -pp/proj1 -idKen:$$ken Y:\productb\archives\proj1
```

Related Topics

For information about...	See...
Setting the archive location	"SetArchiveLocation command" on page 149

GetConfigFile command

List the name and location of configuration files Use the GetConfigFile command to list the path and name of the configuration file for a project database, project, or folder. You can use the **-a** option to list all the configuration files in use by the project database or project, not just the one directly associated with the current project.

Privileges required None.

Alias GCF

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax **GetConfigFile** [*options*] [*entity*]

Where *entity* specifies a project/folder from which to get the value of the configuration file. Note that you only need to specify an entity if want to get the configuration file for an entity other than the current project database or project/folder.

NOTE The current project/folder is set using either the **-pp** option (as described below) or the PCLI_PP variable. A project database is set by using either the **-pr** option (as described below) or the PCLI_PR variable.

Options

To substitute variables and use list files for additional options, see "Command Options" on page 16 for more information.

-a Lists all the configuration file paths that are used by the project, not just the one directly associated with the project you specify. For example, if you specify a project that has a project configuration file associated with it, the PCLI will return the project configuration file and the configuration file associated with the project database in which the project resides.

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

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id**:*PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-**

idCAC_LOGIN. Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp **-pp***project_path*

Sets the current project or folder for this command execution. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Sets the current project database for this command execution. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If no project database is specified, the PCLI_PR variable is used. If it is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

Examples **Example 1:** The following example returns the path and name of the configuration file for the project database located in H:\samples. Note that you do not need to specify an entity because the project database is set using the **-pr** flag.

```
pcli getconfigfile -prH:\samples
Y:\Samples\archives\basecfg.cfg
```

Example 2: The following example returns all configuration files used by the bridge project of the project database located in D:\producta.

```
pcli gcf -prD:\producta -pp/bridge -a
D:\producta\archives\cakos043.cfg
D:\producta\archives\bridge\bridge.cfg
```

Related Topics

For information about...	See...
Setting the configuration file	"SetConfigFile command" on page 151

GetWorkLocation command

List workfile locations for the specified entity

Use the GetWorkLocation command to list the workfile location for the specified project database, project, folder, or versioned file. By default, the workfile location that is returned is the workfile location set in the user's default workspace for the entity specified. You can specify a different workspace from which to get the workfile location by using the **-sp** flag.

For project databases and versioned files, the command always returns the full workfile location.

For projects, you can choose to return either the full path to the workfile location (this is the default if the **-w** flag is not specified) or only the portion of the workfile location that is

stored in the project. This partial path is relative to the project's parent workfile location. For more information on how Version Manager stores relative paths, see the *User's Guide.*

Privileges required: None.

Alias: GWL

Exit codes:

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax: **GetWorkLocation** [*options*] [*entity*]

Where *entity* specifies a project/folder or versioned file for which to get the workfile location. Note that you only need to specify an entity if want to get the workfile location for an entity other than the current project database or project/folder.

NOTE The current project/folder is set using either the -pp option (as described below) or the PCLI_PP variable. A project database is set by using either the -pr option (as described below) or the PCLI_PR variable.

Options

To substitute variables and use list files for additional options, see "Command Options" on page 16 for more information.

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

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id**:*PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-**

idCAC_LOGIN. Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp **-pp***project_path*

Sets the project or folder for this command execution. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Sets the project database for this command execution. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If no project database is specified, the PLCI_PR variable is used. If it is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

-sp **-sp**{[/@/*userID*] | [Public/*parent_workspace*/*child_workspace*]}

Specifies the workspace from which to return the workfile location. If not specified, the workfile location is returned from the user's default workspace. Note that user IDs are case-sensitive. To specify the Root Workspace, enter /@/RootWorkspace for the workspace value.

The specified workspace must already exist. Use the CreateWorkspace command to create a new workspace.

-w Returns only the portion of the workfile location stored within the specified entity. This option also allows you to see the string that was set by the SetWorkLocation command to determine if the workfile location is relative to the parent or is absolute.

Examples **Example 1:** The following example lists the workfile location for the Prj1 project in the Samples project database. The location is taken from the user's default workspace because no workspace is specified (**-sp** option). Note that a value of /Prj1 is specified for the entity argument. The project could have been set using the **-pp** flag or the PCLI_PP variable.

```
pcli getworklocation -prH:\Samples /Prj1 Z:\samples\work\Prj1
```

Example 2: The following example returns only the actual value that is associated with the project or versioned file. Unless this value already contains a fully qualified path, it will be appended to the workfile location of the parent project whenever a fully qualified path must be calculated (e.g. Get, or GetArchiveLocation without **-a**).

```
pcli getworklocation -w /Prj1
Prj1
```

Or

```
pcli getworklocation -w /
Z:\samples\work
```

Example 3: The following example lists the workfile location for the same project using the workfile location set in a private workspace named dev. Note that in this example the Prj1 project is set using the **-pp** flag.

```
pcli getworklocation -prH:\Samples -pp/Prj1 -sp/@/annab/dev
D:\Mywork\Project1
```

Related Topics

For information about...	See...
Setting the workfile location	"SetWorkLocation command" on page 155

If command

Use If to execute command sets based on boolean results

Use the If command in the same manner as the standard C-like conditional. This command allows scripts to execute one set of commands or another depending on the result of a boolean command. A boolean command is a command that returns a status of true or false, where true is 0 (zero) and false is any other status.

Typically, the boolean command used is the Test command, which evaluates boolean expressions, such as string comparisons. See the "Test command" on page 160 for more information.

Privileges required

None.

Alias

None.

Exit codes

Returns the value of the last command executed in the If statement when successful.

-2 PCLI command not found
-3 A non-PCLI related error or a command-specific error
-6 An invalid argument was specified
-7 An argument for a flag that is not needed
-8 A missing argument for a flag
-9 Wrong type was specified for an option's argument
-10 The specified file name cannot be read
-11 A required argument is missing
-12 A security exception occurred
-13 An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax

```
if [options] boolean_command
{
   action. . .
}
[elif boolean_command
{
   action. . .
}. . .]
[else
{
   action. . .
```

```
}]
```

NOTE The else and elif clauses are optional, but the else clause must be the last clause.

Where:

- *boolean_command* specifies the name of the boolean command to use, typically the PCLI Test command.
- *action* specifies the action to take. This action can be a PCLI command or any external program. Place each action on a separate line.

Options

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

Example The following example sets the Configfile variable to the output of the GetConfigfile command execution. Then, the script looks at the value of the variable and compares it to a string (a configuration file path and name). If the value of the variable matches the string, the If statement is exited; if not, the SetConfigFile command is executed:

```
set -vConfigfile $[getconfigfile -prD:\producta]
If Test $Configfile=H:\producta\archives\cb09zjk.cfg
{
   exit
}
else
{
   setconfigfile -prD:\producta -cH:\producta\archives\cb09zjk.cfg
}
```

Related Topics

For information about...	See...
Boolean command	"Test command" on page 160
Setting variables	"Set command" on page 142
Other commands used in scripts	ArraySize on page 52, Break on page 54, Calc on page 55, Continue on page 57, Echo on page 63, Exit on page 73, For on page 76, Readline on page 135, Return on page 138, Run on page 139, Test on page 160, and While on page 175.

ImportArchives command

Import individual archives and recursively add directories

Use the ImportArchives command to add versioned files to a project database, project, or folder from an archive location that you specify. You can add a directory tree, a single

directory, or individual files to a project database or project. Version Manager creates a versioned file that references the original archive for each archive you import.

NOTE On some UNIX systems, the default open file descriptor limit may be set too low. We recommend that you set your file descriptor limit to 128 or higher. For very large databases, we recommend setting the limit as high as allowed by the operating system.

By default, when you import archives to a project database, a project, or a 5.3/6.0 folder, the directory from which you import the archives is used as the archive location unless you specify the **-c** option. This option sets the archive location to the archive location defined for the project database or project to which they were imported.

NOTE When you import archives to a 5.3/6.0 project, your only option is to copy the archives to the project's archive location. In the Version Manager desktop client, you can select the archive location. In the PCLI, however, you cannot select the archive location; the PCLI uses the first archive location specified for the 5.3/6.0 project.

When you import archives to a project database or project, you must specify an archive directory or list of archives. You can also use the command to recursively add projects based on the file structure of the archive directories. The PCLI creates a project with the same name as each directory, and in this project, creates a versioned file for each archive you import. If a versioned file with the same name as an archive you are importing already exists in the project, the existing versioned file is **not** overwritten. When you recursively add subprojects, the archive locations of the new projects are appended to the archive location of the parent project.

By default, if a versioned file with the same name as an archive you are importing already exists in the project, the PCLI will display a warning message. You can turn off this warning message by using the **-qw** option.

If you choose to copy the archives to the project's archive location (the **-c** option), by default, the PCLI will display a warning message if an archive with the same name as one you are importing already exists in the project's archive location. You can turn off this warning message by using the **-qc** option along with the **-c** option. In either case, the existing archive is not overwritten.

NOTE To make the versioned files visible in the desktop client, select the project and then select View | Refresh.

Privileges required

Add or Remove Versioned Files. For more information on privileges, see the Administrator's Guide.

Alias

`IA`

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax **ImportArchives** [*options*] {[*archive_directory*] | [*archive_file ...*]}

Where:

archive_directory specifies the directory that contains the archive files to import. When you specify a directory, a project is created and versioned files are added to the project.

archive_file specifies the names of the individual archives to import.

NOTE You can specify more than one directory or archive file, or a combination of the two, in the command line. Separate multiple values with a blank space.

Options

To substitute variables and use list files for additional options, see "Command Options," for more information.

-c Copies the archives to the project's archive location. If an archive with the same name as one you are importing already exists in the project's archive location, the PCLI displays a warning. The existing archive is not overwritten. To turn off the warning message, use the **-qc** option.

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

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp **-pp***project_path*

Specifies the project or folder to which you want to import the archives. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Specifies the project database to which you want to import the archives. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If no project database is specified, the PLCI_PR variable is used. If it is not defined, then an error message is displayed and the command is aborted.

-qc Turns off the warning message that is displayed when you use the **-c** option. This option is valid only when used with the **-c** option.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

-qw Turns off the warning message that is displayed if a versioned file with the same name as an archive you are importing already exists in the project.

-sp **-sp**{[/@/*userID*] | [Public/*parent_workspace*/*child_workspace*]}

Specifies a public or private workspace. The workfile location of the workspace is used as the workfile location for the files you are adding. Note that user IDs are case-sensitive. To specify the Root workspace, enter `/@/RootWorkspace` for the workspace value.

The specified workspace must already exist. Use the CreateWorkspace command to create a new workspace.

-z Includes archives in subdirectories.

Example **Example 1:** The following example imports the archive file difrpt.gif-arc to the Parcheesi project. The file is copied to the project's archive location (**-c** option).

```
pcli ImportArchives -prH:\BoardGames -pp/Parcheesi -c H:\Source\Parcheesi\difrpt.gif-arc
```

Example 2: The following example imports all archive files in and below `H:\Source\Parcheesi` to the /Parcheesi project. The file is copied to the project's archive location (**-c** option). The second instance creates the project Parcheesi, if it does not already exist, under the project specified by **-pp** because the argument passed is a directory.

```
pcli ImportArchives -prH:\BoardGames -pp/Parcheesi -c -z H:\Source\Parcheesi\*
```

Or

```
pcli ImportArchives -prH:\BoardGames -pp/ -c -z H:\Source\Parcheesi
```

Related Topics

For information about...	See...
Creating projects	"CreateProject command" on page 58
Creating project databases	"CreateProjectDB command" on page 61
Adding workfiles	"AddFiles command" on page 46

ImportPDB command

Use output generated by ExportPDB to create a new database

Use the ImportPDB command to import project database and project information from an INI file format created by the ExportPDB command. Note that workfiles, configuration files, access control database files, and archives are not imported, but references to these files are imported unchanged.

If the information in the INI file is from a Version Manager project database, then a new project database is created at the specified location and all project data is imported.

If the information in the INI file is from a Version Manager 5.3/6.0 project root, then a Version Manager project database into which to import the 5.3/6.0 information must already exist. When a 5.3/6.0 project root is exported, only users and workspaces are exported.

Privileges required

None.

Alias

`IPDB`

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax

ImportPDB [*options*] **-f***input_file*

Where *input_file* names the file that contains the exported project database information. The **-f** option is required.

Options

To substitute variables and use list files for additional options, see "Command Options" on page 16 for more information.

-f **-f***input_file*

Names the file that contains the exported project database. This is required.

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

-pa **-pa***password*

Specifies the password required to create a project database **if** your Administrator has restricted the creation of project databases. In the case where the information in the INI file is from a Version Manager project database, you may need a password in order for Version Manager to create the new project database.

-pr **-pr***project_database*

Specifies the location of the project database that will be created upon command execution for project database information. If you do not specify a project database with the **-pr** option, then the project database location specified in the input file is used. For 5.3/6.0 project root information, this option specifies the location of the existing project database in which to import the information. Note that this command ignores the value of the PCLI_PR variable, if defined.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

Examples **Example 1:** The following example uses the file generated by the ExportPDB command to create a Version Manager project database at the location H:\SampleImport:

```
pcli ImportPDB -fsampleout.txt -prH:\SampleImport
```

Example 2: The following example first creates a project database in which to import Version Manager 5.3/6.0 information, and then imports the 5.3/6.0 information into the newly created project database. Note that only users and workspaces are imported:

```
pcli CreateProjectDB -prH:\Imported60 -nImported60
pcli ImportPDB -prH:\Imported60 -fvm60planet.txt
```

Related Topics

For information about...	See...
Exporting project databases	"ExportPDB command" on page 74
Listing project databases	"ListProjectDB command" on page 116

IsDifferent command

Use the IsDifferent command to test for differences between workfiles, a workfile and a versioned file, or between revisions of versioned files.

Privileges required Check Out Tip, and Check Out Non-Tip. For more information on privileges, see the Administrator's Guide.

Alias `ID`

Exit codes

0	The workfiles and/or revisions are different
1	The workfiles and/or revisions are identical
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing

-12 A security exception occurred
-13 An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax **IsDifferent** [*options*] [*reference_entity*] [*target_entity*]

Where *reference_entity* and *target_entity* specify the versioned files to compare.

Options

-b Blankspaces. Causes the comparison to ignore whitespace.

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

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp **-pp***project_path*

Sets the current project for this command execution. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Sets the current project database for this command execution. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If it is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

-r **-r**[*reference*][:*target*]

Specifies the revision number of the reference and/or target revisions you want to compare.

-sp **-sp**{[/@/*userID*] | [Public/*parent_workspace*/*child_workspace*]}

Specifies a public or private workspace. Note that user IDs are case-sensitive. To specify the Root Workspace, enter /@/RootWorkspace for the workspace value. If a workspace is not specified, the user's default workspace is used.

-v **-v**[*reference*][:*target*]

Specifies the version label of the reference and/or target revisions you want to compare.

-wr **-wr**[*reference_workfile*]

Specifies the reference workfile you want to compare.

-wt **-wt**[*reference_target*]

Specifies the target workfile you want to compare.

Examples **Example 1:** The following example shows how a PCLI script can compare the default revision of a versioned file against the workfile in the active workspace.

```
# Set project database
Set -vPCLI_PR "/usr/serena/vm/common/sampledb"
# Set user ID
Set -vPCLI_ID "Admin"
Set -vFile "/chess/server/server.bat"
Set -vWorkDir $[GetWorkLocation "$File"]

IsDifferent -wt"$WorkDir" "$File"

# Store the exit code of the previous command
Set -vRC $?

If test $RC = 0
{
   Echo "Default revision of $File is different than
     workfile in $WorkDir"
}

ElIf test $RC = 1
{
   Echo "Default revision of $File is identical to workfile
     in $WorkDir"
}

Else
{
   Echo "There was a problem running IsDifferent on $File"
}
```

Example 2: The following example shows how a PCLI script can compare two revisions specified by labels in all versioned files from a particular project down (recursively).

```
# Set project database
Set -vPCLI_PR "C:\Users\All Users\Serena\VM\SampleDb"
# Set user ID
```

```
Set -vPCLI_ID "admin"
Set -vTopLevelProject "/chess"
Set -vReferenceLabel "REL 1.0"
Set -vTargetLabel "LATEST"
# Use "ListVersionedFiles -l -z" to obtain a list of fully
   qualified entity paths ("/project/subproject/versioned file")
For File in $[ListVersionedFiles -l -z "$TopLevelProject"]
{
   # Need to specify the $File entity twice (source and
      target)
   If Is Different -v"$ReferenceLabel:$TargetLabel" "$File"
      "$File"
      {
      Echo "Revisions $ReferenceLabel and $TargetLabel are different in
          $File"
      }
}
```

Related Topics

For information about...	See...
Listing the location of workfiles	"GetWorkLocation command" on page 88
Listing the versioned files in a project	"ListVersionedFiles command" on page 122
Using PCLI commands in scripts	Chapter 2, "Scripting" on page 33

Label command

Use the Label command to assign a version label to a revision of the specified versioned files.

Privileges required: Add Version Label, and Modify Version Label. For more information on privileges, see the Administrator's Guide.

Alias: None.

Exit codes:

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax: **Label** [*options*] **-v***version entity ...*

Where *entity* specifies the folder, project, project database, or versioned file that you want to label and *version* specifies the version label.

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

Options

-f Floating. Specifies that the label will float to the newest revision.

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

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp **-pp***project_path*

Sets the current project for this command execution. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Sets the current project database for this command execution. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If it is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qr Ignores revisions, labels, and groups that are not located in the archives. Does not print warning messages or set a non-zero exit code.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

-r **-r**[*revision*]

Specifies the revision of the versioned files you want to label. If you don't use this option, the label is assigned to the default revision; if no default revision is in effect, the label is assigned to the tip revision.

-sp **-sp/@/***userID* | **Public**/*parent_workspace*/*child_workspace*

Specifies a public or private workspace. Note that user IDs are case-sensitive. To specify the Root Workspace, enter **/@/**RootWorkspace for the workspace value. If a workspace is not specified, the user's default workspace is used.

-v **-v**[*version*]

Specifies the version label you want to assign. This attribute is required.

-z Includes versioned files in subprojects.

Examples **Example 1:** The following example shows two ways to assign the label *Release 1.23* to revision 1.7 of the file `server.bat`, which is located in the project /chess/server.

```
pcli Label -prH:\VMSampleDb -idAdmin -r1.7 -v"Release 1.23" /chess/server/server.bat
pcli Label -prH:\VMSampleDb -idAdmin -r1.7 -v"Release 1.23" -pp/chess/server server.bat
```

Example 2: The following example assigns the version label LATEST to the default revision (or to the tip if no default is defined) of all versioned files in the entire project database.

```
pcli Label -pr/usr/serena/vm/common/sampledb -idAdmin -f -vLATEST -z /
```

Related Topics

For information about...	See...
Deleting version labels	"DeleteLabel command" on page 68
Listing floating version labels	"ListFloatingLabels command" on page 114

List command

List child entities and additional attributes Use the List command to list entities, their attributes, and their children, such as the names and attributes for all entities within a project database. This is demonstrated in the Syntax 1 section below.

List can also return the valid entity names or valid attribute names for the specified project database. This is shown in the Syntax 2 and Syntax 3 sections below.

Privileges required None.

Alias None.

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument

-10 The specified file name cannot be read
-11 A required argument is missing
-12 A security exception occurred
-13 An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax 1 **List** [*common_options*] [*filtering_options*] [*formatting_options*] [*navigation_options*] [*entities*]

Syntax 2 **List** [*common_options*] [**-t***entity_type*]...] [[**-nt***entity_type*]...] **-it**

Syntax 3 **List** [*common_options*] [**-t***entity_type*]...] [[**-nt***entity_type*]... [**-a**[*entity_type*][:*attribute_type*]]... [**-ca**[*entity_type*][:*attribute_type*]]... [**-na**[*entity_type*][:*attribute_type*]]... **-ia**

Common Options for All Syntaxes

To substitute variables and use list files for additional options, see "Command Options" on page 16 for more information.

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

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp **-pp***project_path*

Specifies the default project or folder to list. This option overrides the value of the value of the PCLI_PP variable for a single command execution. For more information, see -pp on page 17.

-pr **-pr***project_database*

Specifies the project database or project root for which you wish to list attributes or information. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

-sp **-sp/@/***userID* | **Public**/*parent_workspace*/*child_workspace*

Specifies a public or private workspace from which to return the values for certain attributes (such as BranchVersion and BaseVersion). Note that user IDs are case-sensitive. To specify the Root Workspace, enter **/@/**RootWorkspace for the workspace value.

The specified workspace must already exist. Use the CreateWorkspace command to create a new workspace.

Options for Syntax 1

Formatting Options

These options control the format of the listed values.

-a **-a**[*entity_type*:][*attribute_type*]. . .

Lists the specified attribute, or all attributes for entities of the given type. This option may be repeated to list several attributes. If this option is not specified, then no attributes are listed (this is the default). Values displayed by **-ca** may include information inherited from both the workspace and the project hierarchy, whereas **-a** does not consider information inherited from the project hierarchy.

If only *attribute_type* is given, for example **-a**ArchiveLocation, then the given attribute is listed for all entities that have it.

If only *entity_type* is given, for example, **-a**Project:, then all attributes are listed for entities of the given type. Note that if the attribute type is not specified, the entity type must be followed by a colon.

If the option is specified with neither *entity_type* or *attribute_type*, then all attributes are listed for all listed entities.

If the user specifies an attribute to be listed for an entity type that would not otherwise have been listed, i.e., **-t**Foo **-a**Bar:Baz, then List treats this as if **-t**Bar had also been specified.

All **-a** and **-ca** flags are processed before all **-na** flags.

This option is not valid with the **-it** or **-ia** options.

-ca **-ca**[*entity_type*:][*attribute_type*]. . .

Lists the computed value of the specified attribute. Use the **-a** option to list the specified attribute.

Both **-a** and **-ca** may be specified for the same attribute; in this case both the local value and the computed value are listed.

For most attribute types **-ca** produces the same output as **-a**. An example of where these might differ is the `Project:ArchiveLocation` attribute, where the computed value can contain inherited information. Values displayed by **-ca** may include information inherited from both the workspace and the project hierarchy, whereas **-a** does not consider information inherited from the project hierarchy. As an example, the noncomputed value might be proj1, and the computed value might be N:\vmdata\archives\proj1.

All **-a** and **-ca** flags are processed before all **-na** flags.

This option is not valid with the **-it** or **-ia** options.

-fx Lists output in XML format and includes an XML header tag. Must be used with the **-a** or **-ca** option. The **-nx** option is the same, except it does not include the XML header. See "XML Output" on page 109.

Lists full entity paths; otherwise only leaf names are listed. When you specify an option that looks recursively into subprojects (such as the **-z** option), the **-l** option suppresses the "Contents of" container headers. For more information, see "Output" on page 107.

-na **-na**[*entity_type:*][*attribute_type*]. . .

Does not list attributes of the given type. This option may be specified multiple times to filter out several types of attributes.

All **-na** flags are processed after all **-a** and **-ca** flags.

-nx Lists output in XML format but omits the XML header tag. Must be used with the **-a** or **-ca** option. The **-fx** option is the same, except it includes an XML header. See "XML Output" on page 109.

Navigation Options

These options allow you to recursively list the children of entities and control which entities List examines. If you do not specify a navigation option, then only the entities specified (directly or with wildcards) are examined; their children are not. Note that you can use multiple navigation options to list multiple entities.

One use of navigation options is to optimize performance. It can be unnecessarily slow to have List navigate through many entities that do not need to be listed.

NOTE These options control which parts of the entity hierarchy are visited by the List command, but just because a particular entity is visited doesn't mean it is printed out–the filtering flags are also taken into account.

-z Recursively lists entities in subprojects. This flag is equivalent to **-zt**`ProjectRoot` **-zt**`Project` **-zt**`Folder`. This option lists the children of project roots and projects; no entities inside the versioned file are listed (i.e., revisions and branches). If a project contains no archives, the project is not listed in the output. The **-zt** flag lists entities below the project level.

This option is not valid with the **-it** or **-ia** options.

-zt **-zt**[*entity_type*]. . .

Recurse through children of entities of the given type. If *entity_type* is not given, recurse through all entity types. If a project contains no versioned files, the project is not listed in the output. This option may be repeated to list multiple entity types.

All **-zt** flags are processed before all **-zn** flags.

This option is not valid with the **-it** or **-ia** options.

-zn **-zn**[*entity_type*]. . .

Do not recurse through children of entities of the given type. This option may be repeated to list multiple entity types.

All **-zn** flags are processed after all **-zt** flags.

This option is not valid with the **-it** or **-ia** options.

Filtering Options

These options control which entities are printed out as List navigates the entity hierarchy.

-i Ignores case in pattern matching (see the **-m** option).

-lr Lists the default revision.

-m **-m***pattern*

Lists only entities whose leaf name matches the given pattern. This option may be given multiple times to match more than one pattern.

-nt **-nt***entity_type. . .*

Does not list entities of the given type. This option may be specified multiple times to filter out several types of entities.

All **-nt** flags are processed after all **-t** flags.

-t **-t***entity_type. . .*

Lists only entities of the given type. This option may be repeated to list several types of entities. If this option is not specified, then entities of all types are listed (this is the default).

All **-t** flags are processed before any **-nt** flags.

Note unvisited entities are never printed. For example, if you specify **-z**, then branches and revisions won't be visited during navigation and therefore will not be listed, even if you specifically say **-t**`Revision` or **-t**`Branch`.

Options for Syntax 2

-it Lists all possible entity types rather than listing the entities themselves.

If **-t** and/or **-nt** are also specified, the PCLI filters which entity types are listed.

-t, -nt See "Filtering Options" on page 106.

Options for Syntax 3

-ia Lists all possible attribute types for the specified entity, rather than listing the entities themselves.

If **-t**, **-nt**, **-a**, **-ca**, and/or **-na** are also specified, they filter which entity and attribute types are listed.

-t, -nt See "Filtering Options" on page 106.

-a, -ca, -na See "Formatting Options" on page 104.

Listing Entity and Attribute Types

As discussed above, entity and attribute types (such as a Project, Workspace, or VersionedFile) can also be listed. Both the attribute types and the entity types are case-sensitive. You must use the proper case in specifying an attribute or an entity.

The following list shows the valid entity types. You can view these entity types using the **-it** option (syntax 2).

Workspace	Group
Folder	User
Project	ProjectRoot
Branch	Revision
VersionedFile	

To list valid attribute types for an entity, use the **-ia** option (syntax 3). Using this **-ia** option without specifying an entity, lists attribute types for all entities.

Output

In general, use the **-l** option if the output will be used in a script. If **-l** is not specified, then output is in a more readable format. In the examples below, output is in bold.

If you do not specify the **-l** option, entities are listed by their leaf names. The asterisk at the end of the command specifies all entities are to be listed. For example:

```
pcli list -prC:\mypdb /*
   Proj1
   Proj2
```

If -l is specified, entities are listed by their full paths. For example:

```
pcli list -prC:\mypdb -l /*
   /Proj1
   /Proj2
```

If **-z** is specified and **-l** is not, container headers are printed, and if a project contains no versioned files, the project is not listed. For example:

```
pcli list -prH:\Samples -z /*
   Contents of /Prj1:
   BRANCH.GIF
   COPY53.PDF
   MKTG.DOC
   Subprj1

   Contents of /Prj2:
   STARTUP.DOC
```

```
   Subprj2
   TARLINK.FM
   TRUNK.GIF

   Contents of /Prj2/Subprj2:
   BRANCH.GIF
   PAYROLL
   PVCSAPP.LOG

   Contents of /test:
   BvTEST.DAT
   CONTROLS.TXT
```

If **-z** is specified with **-l**, container headers are not printed and all projects are listed whether or not they contain versioned files (unlike the previous example that uses only the **-z** option). For example:

```
pcli list -prC:\mypdb -l -z /Proj1
   /Proj1
   /Proj1/foo.txt
   /Proj1/Subproj
   /Proj1/Subproj/bar.txt
```

If **-a** is specified, an extra blank line is output before each entity name, and the specified attribute(s) is printed afterward in the format *attribute name*=*attribute value*. For example:

```
pcli list -prC:\mypdb -a /*
   Proj1
   EntityType=Project
   WorkPath=Proj1
   etc...

   Proj2
   EntityType=Project
   WorkPath=Proj2
   etc...
```

If **-ca** is specified, an extra blank line is output before each entity name and the specified attribute(s) is printed afterward in the format *attribute name*=*computed value*. For example:

```
pcli list -prC:\mypdb -ca /*
   Proj1
   EntityType=Project
   WorkPath=C:\work\Proj1
   etc...

   Proj2
   EntityType=Project
   WorkPath=C:\work\Proj2
   etc...
```

If both **-a** and **-ca** are specified, the noncomputed value for each attribute is printed first, immediately followed by the computed value. For example:

```
pcli list -prC:\mypdb -a -ca /*
   Proj1
```

```
   EntityType=Project
   EntityType=Project
   WorkPath=Proj1
   WorkPath=C:\work\Proj1
   etc...

   Proj2
   EntityType=Project
   EntityType=Project
   WorkPath=Proj2
   WorkPath=C:\work\Proj2
   etc...
```

The **-it** option prints entity type names one per line. For example:

```
pcli list -prC:\mypdb -it
   ProjectRoot
   Project
   User
   etc...
```

The **-ia** option prints attribute type names one per line in the format used by the **-a**, **-ca**, and **-na** flags. For example:

```
pcli list -prC:\mypdb -ia
   ProjectRoot:EntityType
   ProjectRoot:WorkPath
   etc...
```

XML Output

Both the **-fx** and **-nx** options produce a list in XML format. **-fx** includes an XML header tag, **-nx** does not.

XML Output Examples

XML Example 1: On a single file:

```
C:\>pcli List -pr"C:\Users\All Users\Serena\VM\SampleDB" -idAdmin -a -
    fx /chess/server/server.bat
PVCS Version Manager (PCLI) v8.6.1.0 (Build 162) for Windows/x86_64
Copyright (C) 1985-2018 Serena Software, Inc., a Micro Focus company.
    All rights reserved.
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE VersionManagerXML SYSTEM "VmExport.dtd">
<VersionManagerXML Version=".9"
Date="Mon Jun 30 13:39:30 GMT-07:00 2008"
PDBRoot="C:\Users\All Users\Serena\VM\SampleDB">
<VersionedFile EntityPath="/chess/server/server.bat"
Name="server.bat"
ArchivePath="C:\Users\All
    Users\Serena\VM\SampleDB\archives\chess\server\server.bat-arc"
ArchiveOwner="Admin"
CreateTime="895448226000"
LockCount="0"
RevisionCount="34"
WorkPath="."
WorkPathNamespace="0">
<WorkfileDescription>Sample Project Database - first revision.
```

```
</WorkfileDescription>
</VersionedFile>
</VersionManagerXML>
```

XML Example 2: On a project using values as stored (**-a**):

```
C:\>pcli List -pr"C:\Users\All Users\Serena\VM\SampleDB" -idAdmin -a -
    nx /bridge
PVCS Version Manager (PCLI) v8.6.1.0 (Build 162) for Windows/x86_64
Copyright (C) 1985-2018 Serena Software, Inc., a Micro Focus company.
    All rights reserved.
<Project EntityPath="/bridge"
Name="bridge"
WorkPath="bridge"
WorkPathNamespace="0"
ArchivePath=""
ArchiveLocation="bridge"
VersionSelection="0"
VersionSelectionLabels=""
VersionSelectionPromoGroup=""
DefaultVersion=""
BaseVersion=""
BranchVersion=""
DefaultDevPromoGroup=""
ConfigFlags="2"
ConfigPath="C:\Users\All
    Users\Serena\VM\SampleDb\archives\bridge\bridge.cfg">
</Project>
```

XML Example 3: On a project using computed values (**-ca**):

```
C:\>pcli List -pr"C:\Users\All Users\Serena\VM\SampleDB" -idAdmin -ca -
    nx /bridge
PVCS Version Manager (PCLI) v8.6.1.0 (Build 162) for Windows/x86_64
Copyright (C) 1985-2018 Serena Software, Inc., a Micro Focus company.
    All rights reserved.
<Project EntityPath="/bridge"
Name="bridge"
WorkPath="C:\Users\All Users\Serena\VM\SampleDb\work\bridge"
WorkPathNamespace="0"
ArchivePath=""
ArchiveLocation="C:\Users\All Users\Serena\VM\SampleDb\archives\bridge"
VersionSelection="0"
VersionSelectionLabels=""
VersionSelectionPromoGroup=""
DefaultVersion=""
BaseVersion=""
BranchVersion=""
DefaultDevPromoGroup=""
ConfigFlags="2"
ConfigPath="C:\Users\All
    Users\Serena\VM\SampleDb\archives\bridge\bridge.cfg">
</Project>
```

XML Example 4: Same as Example 3, but recursive on all files in the project:

```
C:\>pcli List -pr"C:\Users\All Users\Serena\VM\SampleDB" -idAdmin -z -
    ca -nx /bridge
```

XML Example 5: On all revisions in a file:

```
C:\>pcli List -pr"C:\Users\All Users\Serena\VM\SampleDB" -idAdmin -zt -
    ca -nx /chess/client/ChessViewer.jpr
PVCS Version Manager (PCLI) v8.6.1.0 (Build 162) for Windows/x86_64
Copyright (C) 1985-2018 Serena Software, Inc., a Micro Focus company.
    All rights reserved.
<VersionedFile EntityPath="/chess/client/ChessViewer.jpr"
Name="ChessViewer.jpr"
ArchivePath="C:\Users\All
    Users\Serena\VM\SampleDB\archives\chess\client\ChessViewer.jpr-arc"
ArchiveOwner="Admin"
CreateTime="895448230000"
LockCount="0"
RevisionCount="2"
WorkPath="C:\Users\All Users\Serena\VM\SampleDb\work\client"
WorkPathNamespace="0">
<WorkfileDescription>Sample Project Database - first revision.
</WorkfileDescription>
<Revision EntityPath="/chess/client/ChessViewer.jpr/1.1"
Name="1.1"
Author="kerstinb"
LockCount="0"
ModificationDate="895492104000"
CheckedInDate="895492124000">
<ChangeDescription>Changed the source path.
</ChangeDescription>
<Label Name="REL_1.0"/>
<FloatingLabel Name="LATEST"/>
</Revision>
<Revision EntityPath="/chess/client/ChessViewer.jpr/1.0"
Name="1.0"
Author="Admin"
LockCount="0"
ModificationDate="895448232000"
CheckedInDate="895448230000">
<ChangeDescription>Initial revision.
</ChangeDescription>
</Revision>
</VersionedFile>
```

Examples

Example 1: To list the relative workfile location attribute of a project:

```
pcli List -prH:\BoardGames -aWorkPath /parcheesi
Parcheesi
WorkPath=work\Parcheesi
```

Example 2: To list the computed workfile location of a project:

```
pcli List -prH:\BoardGames -caWorkPath /parcheesi
Parcheesi
WorkPath:=H:\BoardGames\work\Parcheesi
```

Example 3: To print the location of a project's data directory:

```
pcli List -prH:\BoardGames -caDirectory /parcheesi
Parcheesi
Directory=H:\BoardGames\Pkzshmj.prj
```

Example 4: To return a recursive listing starting at the root and printing out the work path attributes of just the projects (not the versioned files):

```
pcli List -prH:\Samples -aProject:WorkPath -l -z /
/
/contents.htm
/Prj1
WorkPath=Prj1
/Prj1/admin_ol.doc
/Prj1/BRANCH.GIF
/Prj1/COPY53.PDF
/Prj1/MKTG.DOC
/Prj1/Subprj1
WorkPath=Subprj1
/Prj1/template.fm
/Prj1/title.fm
/Prj2
WorkPath=Prj2
/Prj2/STARTUP.DOC
/Prj2/Subprj2
WorkPath=Subprj2
/Prj2/Subprj2/BRANCH.GIF
```

Example 5: To list all public workspaces in a project database:

```
pcli List -prC:\mypdb -zt -tWorkspace /@/Public
Contents of /@/Public:
annab
Documentation
Contents of /@/Public/annab:
Contents of /@/Public/Documentation:
```

Example 6: To list the attributes that are available for the BoardGames project database:

```
pcli list -prH:\BoardGames -a
BoardGames
EntityType=ProjectRoot
WorkPath=H:\BoardGames
Name=BoardGames
ArchiveLocation=H:\BoardGames\archives
BranchVersion=null
BaseVersion=null
DefaultVersion=null
VersionSelectionPromoGroup=null
VersionSelectionLabels=null
VersionSelection=0
ArchivePath=null
ConfigFlags=3
ConfigPath=H:\BoardGames\archives\cn8gs971.cfg
ToolbarConfigPath=null
WorkPathNamespace=0
```

Example 7: The following example lists the contents of a Version Manager 5.3/6.0 project root:

```
pcli list -prH:\vm60planet -z -ztFolder /
Contents of /:
Excel
```

```
Girl
Master Project
Volley
Word
Contents of /Excel:
Copernicus
Contents of /Excel/Copernicus:
Check_In.xls
checkout.xls
FRM8.tmp
FRM9.tmp
FRMA.tmp
FRMB.tmp
Contents of /Girl:
Contents of /Volley:
Contents of /Word:
Copernicus
Contents of /Word/Copernicus:
65Install.doc
INSTALL.DOC
```

Example 8: The following example lists the labels for revision 1.2:

```
pcli list -prC:\Users\All Users\Serena\VM\sampledb -idadmin -
    aArchiveRevision:Labels
   /chess/client/images/cmpieces.gif/1.2
1.2
ArchiveRevision:Labels=[REL_1.0]
```

Example 9: The following example lists the labels for revision 1.2.1.0 on the branch1.2.1:

```
pcli list -prC:\Users\All Users\Serena\VM\sampledb -idadmin -
    aArchiveRevision:Labels
   /chess/client/images/cmpieces.gif/1.2/1.2.1/1.2.1.0
1.2.1.0
ArchiveRevision:Labels=[]
```

Example 10: The following example shows how to list multiple files including revisions under the client project using the **-zt** option:

```
pcli list -prC:\Users\All Users\Serena\VM\sampledb -idadmin -zt -
    aArchiveRevision:Labels
   /chess/client/*
```

(The above command would give a listing of all files, revisions, and labels under the client project.)

Example 11: The following example lists all the branches on versioned files in the project database:

```
pcli list -prC:\Users\All Users\Serena\VM\sampledb -idadmin -zt -
    tBranch /
```

Example 12: The following example lists all the labels on versioned files in a project database:

```
pcli list -prC:\Users\All Users\Serena\VM\sampledb -idadmin -
    caArchiveRevision:Labels
   -caArchiveRevision:FloatingLabels -ztVersionedFile -z -l /
```

Related Topics

For information about...	See...
Listing versioned files	"ListVersionedFiles command" on page 122
Listing project databases	"ListProjectDB command" on page 116

ListFloatingLabels command

Use the ListFloatingLabels command to list the floating labels for specified entities.

Privileges required: View Archive Revisions. For more information on privileges, see the Administrator's Guide.

Alias: LFL

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax: **ListFloatingLabels** [*options*] [*entity*]

Where *entity* specifies the folder, project, project database, or versioned file you want to act on.

Options

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

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by

your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id**:*PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-l Lists full entity paths, including the revision number ("/project/versioned_file/revision#"); otherwise only the revision number is listed.

-pp **-pp***project_path*

Sets the current project for this command execution. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Sets the current project database for this command execution. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If it is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

-sp **-sp/@/***userID* | **Public**/*parent_workspace*/*child_workspace*

Specifies a public or private workspace. Note that user IDs are case-sensitive. To specify the Root Workspace, enter **/@/**RootWorkspace for the workspace value. If a workspace is not specified, the user's default workspace is used.

-z Recursively lists entities in subprojects.

Examples **Example 1:** The following example lists all floating labels in the project database in a format that includes the entity path with the revision number ("/project/versioned_file/revision#").

```
pcli ListFloatingLabels -prC:\Users\All Users\Serena\VM\SampleDb -idAdmin -z -l /
```

Example 2: The following example lists all floating labels in the project /bridge, but does not include its subprojects.

```
pcli ListFloatingLabels -prC:\Users\All Users\Serena\VM\SampleDb -idAdmin /bridge/*
```

Related Topics

For information about...	See...
Deleting version labels	"DeleteLabel command" on page 68
Adding version labels	"Label command" on page 100

ListProjectDB command

List active and home project databases

Use the ListProjectDB command to list:

- The current project database location.
- All the active project database locations currently used by the Version Manager desktop client on the current machine. These project database locations are saved by the desktop client in the islv.ini file, and would be reopened in the desktop client the next time you started Version Manager.
- The project databases that are found in the file system directories you specify.

Privileges required

None.

Alias

`LPDB`

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax

`ListProjectDB` [**`-a`**] [**`-t`**]

or

`ListProjectDB` [**`-z`**] [**`-t`**] *`directory_location. . .`*

Where *`directory_location`* specifies the file system directories in which to search for project databases. Separate multiple values with a blank space.

Options

To substitute variables and use list files for additional options, see "Command Options" on page 16 for more information.

-a

Lists all the project database locations currently used by the Version Manager desktop client on the current machine or for the current user on UNIX systems.

-h

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

-id

`-id`*`user id`*[: *`user_password`*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **`-id`** to specify your user ID—and password too if a password is assigned to your user ID. For example, **`-id`***`UserID`*: *`Password`*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by

your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-fp **-fp***[file server host]*

Lists all the project databases that are published by the file servers this machine connects to. You can specify this option multiple times to allow output filtering for more than one server.

`[file server host]`

Specifies an optional file server hostname or "hostname:port" pair that limits the output only to this file server. An empty file server host value (or just specifying -fp) displays project databases from all file servers the client connects to.

Example with multiple servers (`http://vmweb:8080/serenafs/FileServer` and `http://or-rgering-w2k8:8080/serenafs/FileServer`).

```
pcli ListProjectDB –fp
\\load
\\japan
X:\
\\vmfs_w2k8\SampleDB
\\vmfs_w2k8\SampleDB_SSO_SBM
\\vmfs_w2k8\SampleDB_SSO_VM

pcli ListProjectDB –fp”vmweb”
\\load
\\japan
X:\
pcli ListProjectDB –fp”vmweb:80”
<nothing>
pcli ListProjectDB –fp”vmweb:8080”
\\load
\\japan
X:\
```

-fv Verbose file server output. Use with -fp to include file server details with every project database. For example:

```
pcli ListProjectDB –fpvmweb -fv
PDB path   : \\load
Path map   : \\load
File Server: http://vmweb:8080/serenafs/FileServer
PDB path   : \\japan
Path map   : \\japan
File Server: http://vmweb:8080/serenafs/FileServer
PDB path   : X:\
Path map   : X:\
```

```
File Server: http://vmweb:8080/serenafs/FileServer
```

-t Lists the type of the project database; `file` means a Version Manager project database, and `file53` means a Version Manager 5.3/6.0 project root.

-z Recursively searches all subdirectories for project databases.

Examples **Example 1:** The following example lists all the project databases currently used by the Version Manager desktop client on the current machine.

```
pcli ListProjectDB -a
```

Example 2: The following example searches recursively for project databases located in H:\ and lists the type of project database along with the path:

```
pcli ListProjectDB -z -t H:\
file   : H:\samples
file53 : H:\vm60Planet
file53 : H:\vm60Planet\Pybk5gd.prj
file53 : H:\vm60Planet\Pf706ha.prj
file   : H:\BoardGames
file   : H:\Stratego
file   : H:\TestLib
file   : H:\testnames
```

Related Topics

For information about...	See...
Listing versioned files	"ListVersionedFiles command" on page 122
Listing other entity attributes	"List command" on page 102

ListPromotionModel command

Use the ListPromotionModel command to list the promotion model that applies to an entity.

Privileges required None.

Alias `LPM`

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax **`ListPromotionModel`** [*`options`*] *`entity`*

Where *entity* specifies a folder, project database, project, or versioned file.

Options

-d Lists development level groups only, rather than all groups.

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

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp **-pp***project_path*

Sets the current project for this command execution. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Sets the current project database for this command execution. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If it is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

-sp **-sp/@/***userID* | **Public**/*parent_workspace*/*child_workspace*

Specifies a public or private workspace. Note that user IDs are case-sensitive. To specify the Root Workspace, enter **/@/**RootWorkspace for the workspace value. If a workspace is not specified, the user's default workspace is used.

Examples **Example 1:** The following example lists the promotion model that applies to the projects /bridge and /chess.

```
pcli ListPromotionModel -prC:\Users\All Users\Serena\VM\SampleDb -idBetsy /bridge/chess
```

Example 2: The following example lists the lowest level promotion groups that apply to all projects and versioned files directly under the project database root (not recursively).

```
pcli ListPromotionModel -prC:\Users\All Users\Serena\VM\SampleDb -
    idDave -d /*
```

Related Topics

For information about...	See...
Listing information for other entities	"List command" on page 102
Listing the contents of project databases	"ListProjectDB command" on page 116

ListRevision command

Use the ListRevision command to list the revision associated with the default version or with a specified version label or promotion group.

Privileges required: View Archive Revisions. For more information on privileges, see the Administrator's Guide.

Alias: LR

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax: **ListRevision** [*options*] *entity* ...

Where *entity* specifies a folder, project database, project, or versioned file.

Options

-a Includes in the output all version labels and promotion group names specified with **-v** and **-g**, even the ones that have no associated revision numbers for a particular versioned file. This option is also required to list files that have no matches for any revision numbers (e.g. when used with **-m**).

-c Climbs the promotion model hierarchy if the specified promotion group does not exist in an archive. Requires the **-g** option.

-d Difference report. Reports if one or more revision numbers are different. If only one version label or promotion group is specified, then reports if it is different from the default version.

Used with the **-n** option, it reports if all the revision numbers are the same.

-f Lists the revision number on the tip of the trunk or branch for floating labels. Without this option, revisions for floating labels are listed as they are assigned (e.g. 1.* or 1.0.1.*).

-g **-g***group*

Lists the revision number associated with the specified promotion group. Note, unlike the Get command, ListRevision will not climb the promotion model hierarchy if there is no exact match for the specified promotion group. However, it will climb the hierarchy when used with the **-c** option.

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

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-m Missing. Reports only if one or more of the specified version labels or promotion groups are not found. Note, to see files where all specified labels and promotion groups are missing, the
-a option is required. Otherwise, only files with at least one match will be shown.

Used with the **-n** option, it reports only if all the specified version labels and promotion groups are found.

-n Reverses the logic of the **-d** and **-m** options. Used with the **-d** option, it reports if version numbers are all the same. Used with the **-m** option, it reports only lines where all version labels or promotion groups are found. Note, this option works only with the **-d** and **-m** options.

-pp **-pp***project_path*

Sets the current project for this command execution. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Sets the current project database for this command execution. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If it is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

-sp **-sp/@/***userID* | **Public**/*parent_workspace*/*child_workspace*

Specifies a public or private workspace. Note that user IDs are case-sensitive. To specify the Root Workspace, enter **/@/**RootWorkspace for the workspace value. If a workspace is not specified, the user's default workspace is used.

-v **-v***version_label*

Lists revisions associated with the specified version label.

-z Recursively lists revisions for versioned files in subprojects.

Examples **Example 1:** The following example lists all files with the version label LATEST and shows the current revision on the tip of the trunk or branch where LATEST is a floating label.

```
pcli ListRevision -prC:\Users\All Users\Serena\VM\sampledb -idartd -f -vLATEST -z /
```

Example 2: The following example lists all files where the labels LATEST and REL_1.0 are not assigned to the same revision. The **-f** option should be used if any of the labels is a floating label.

```
pcli ListRevision -prC:\Users\All Users\Serena\VM\sampledb -idartd -d -f -vLATEST -
vREL_1.0 -z /
```

Example 3: The following example lists all files where the label REL_1.0 is not assigned to the default version (the default behavior when you use **-d** with only one label). Because every file has a default version, any file not having the label will also be included in the output, even though the **-a** option was not used (the default version registers as a match).

```
pcli ListRevision -prC:\Users\All Users\Serena\VM\sampledb -idartd -d -vREL_1.0 -z /
```

Example 4: The following example lists all files located in and below the bridge project that do not have a revision associated with the promotion group QA_Test1.

```
pcli ListRevision -prC:\Users\All Users\Serena\VM\sampledb -idartd -m -gQA_Test1 -a -z /
bridge
```

Related Topics

For information about...	See...
Listing information for other entities	"List command" on page 102
Listing the contents of project databases	"ListProjectDB command" on page 116

ListVersionedFiles command

List attributes of versioned files

Use the ListVersionedFiles command to list:

- Versioned files contained in Version Manager project databases or projects.

- Versioned files contained in Version Manager 5.3/6.0 projects and folders. To list versioned files in 5.3/6.0 projects, specify, for example:

  ```
  pcli lvf -prD:\60prjdb -l -z /ProjectFolder/proj1/*
  ```

  You must specify the * to list the versioned files of a 5.3/6.0 project because versioned files are not typically directly beneath 5.3/6.0 projects but instead inside folders of 5.3/6.0 projects. Therefore, the -z option won't list versioned files of 5.3/6.0 projects as it would versioned files of 5.3/6.0 folders and projects.

NOTE `/ProjectFolder` is a keyword and must be typed exactly as it appears. It is *not* a place holder or part of an actual path on your system.

- The archive paths associated with versioned files (`-a` option).
- The workfile location associated with versioned files (`-w` option).
- Both the archive and workfile locations associated with versioned files (`-aw` option).
- The full path of each versioned file listed (`-l` option).

By default, the command lists only the names of the versioned files with no path, for example, `readme.txt`.

NOTE On some UNIX systems, the default open file descriptor limit may be set too low. We recommend that you set your file descriptor limit to 128 or higher. For very large databases, we recommend setting the limit as high as allowed by the operating system.

Privileges required: None.

Alias: LVF

Exit codes:

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax: **`ListVersionedFiles`** [*options*] [*entity*]

Where *entity* specifies a project/folder for which to list versioned files. Note that you only need to specify an entity if want to list versioned files for an entity other than the current project database or project/folder.

NOTE The current project/folder is set using either the `-pp` option (as described below) or the PCLI_PP variable. A project database is set by using either the `-pr` option (as described below) or the PCLI_PR variable.

Options

To substitute variables and use list files for additional options, see "Command Options" on page 16 for more information.

-a Lists the full archive path, including the file name, instead of just the entity path. If you use this option, you cannot specify **-l**, **-aw**, or **-w**.

-aw Lists the versioned files using the Version Manager 6.0 command-line style format of archive path(work path). For example:

```
"D:\producta\archives\bridge\resrc.h-arc(D:\producta\work\bridge\resrc.h)"
```

Each output line is enclosed in double quotation marks to prevent conflicts that can occur in many command shells.

If you use this option, you cannot specify **-a**, **-l**, or **-w**.

NOTE If your path contains parentheses and you are passing the output of this command to a CLI command, set the PVCS_LEFT_SEPARATOR and PVCS_RIGHT_SEPARATOR environment variables. This allows you to specify a character other than parentheses to enclose the work path.

For information on setting these variables, see the *Command-Line Reference Guide.*

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

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-l Returns the full entity path of the file, for example, `/prj1/subprjA/readme.txt`. Omitting this option lists only leaf names, for example, `readme.txt`. If you use this option, you cannot specify **-a**, **-aw**, or **-w**.

-pp **-pp***project_path*

Specifies the current project or folder for which to list versioned files. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Specifies the project database for which to list versioned files. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If no project database is specified, the PLCI_PR variable is used. If it is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

-sp **-sp/@/***userID* | **Public**/*parent_workspace*/*child_workspace*

Specifies a public or private workspace. Use this option with the **-aw** or **-w** options to return the workfile location stored in a public or private workspace. Note that user IDs are case-sensitive. To specify the Root Workspace, enter **/@/**RootWorkspace for the workspace value.

The specified workspace must already exist. Use the CreateWorkspace command to create a new workspace.

-w Lists the full workfile location, including the file name, instead of just the entity path. If you use this option, you cannot specify **-a**, **-aw**, or **-l**.

-z Recursively lists all subprojects.

Examples **Example 1:** The following example lists the versioned files contained in the Prj1 project in the project database located in H:\Samples:

```
pcli ListVersionedFiles -prH:\Samples /prj1
admin_ol.doc
BRANCH.GIF
COPY53.PDF
MKTG.DOC
template.fm
title.fm
```

Example 2: The following is an example of a recursive listing of all files in the project database located in H:\BoardGames. The versioned file list is returned in the Version Manager 6.0 command-line style format of archive(workpath):

```
pcli ListVersionedFiles -prH:\BoardGames -aw -z
"H:\BoardGames\archives\Parcheesi\14KAPSU.JPG-arc
    (H:\source\sourcefiles\Parcheesi\14KAPSU.JPG)"
"H:\BoardGames\archives\Parcheesi\17BAYC.JPG-arc
    (H:\source\sourcefiles\Parcheesi\17BAYC.JPG)"
etc...
```

Example 3: The following example returns a recursive listing of the full entity paths of the versioned files contained in the Prj2 project in the project database located in H:\Samples:

```
pcli ListVersionedFiles -prH:\Samples -l -z /prj2
/Prj2/STARTUP.DOC
/Prj2/Subprj2/BRANCH.GIF
/Prj2/Subprj2/PAYROLL
/Prj2/Subprj2/PVCSAPP.LOG
/Prj2/TARLINK.FM
```

Related Topics

For information about...	See...
Listing information for other entities	"List command" on page 102
Listing the contents of project databases	"ListProjectDB command" on page 116

Lock command

Use the Lock command to lock a revision of the specified version files.

Privileges required: Lock Tip, and Lock Non-Tip. For more information on privileges, see the Administrator's Guide.

Alias: None.

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax: **Lock** [*options*] [*entity*] ...

Where entity specifies a folder, project database, project, or versioned file.

Options

-g **-g***group*

Specifies the promotion group to which to assign the locked revision. If this option is omitted, the default promotion group is assigned to the checked out revision, if one exists.

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

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by

your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-nb No branching. This option prevents you from locking a revision if a lock would result in a branch upon check in. This option is ignored if the BranchWarn directive is not in effect.

-nm No multilock. This option overrides the MultiLock directive and prevents you from applying another lock to a revision that is already locked.

-pp **-pp***project_path*

Sets the current project for this command execution. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Sets the current project database for this command execution. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If it is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

-r **-r***revision*

Specifies the revision, promotion group, or version to act upon.

-sp **-sp/@/***userID* | **Public**/*parent_workspace*/*child_workspace*

Specifies a public or private workspace. Note that user IDs are case-sensitive. To specify the Root Workspace, enter **/@/**RootWorkspace for the workspace value. If a workspace is not specified, the user's default workspace is used.

-yb Yes branching. Overrides the BranchWarn directive to allow you to lock a revision even if that will result in a branch upon check in.

-ym Yes to multilock. If the MultiLock directive is in effect, this option will apply another lock without first prompting for confirmation.

-z Includes revisions in subprojects.

Examples **Example 1:** The following example locks all files in--but not below--the bridge project and associates them with the lowest level promotion group, Development.

```
pcli Lock -prC:\Users\All Users\Serena\VM\sampledb -idartd -gDevelopment /bridge
```

Example 2: The following example locks the revisions associated with the version label REL_1.0 on all files in and below the chess project, except where this would create a branch.

```
pcli Lock -prC:\Users\All Users\Serena\VM\sampledb -idartd -rREL_1.0 -nb -z /chess
```

Related Topics

For information about...	See...
Getting revisions	"Get command" on page 79
Unlocking revisions	"Unlock command" on page 162

Move command

Use the Move command to move projects or versioned items into a destination project or folder within the originating project database.

Privileges required

Depending on the type of entity you are operating on , you must have the following privileges:

Entity Type	Privileges Required
Versioned file	Add or Remove Versioned Files
Project	Create Project Copy Project Delete Project

For more information on privileges, see the Administrator's Guide.

Alias

None.

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax

Move [*options*] **-d***destination entity ...*

Where:

destination specifies the project or folder to which to move an entity.

entity specifies the project or versioned file you want to move.

NOTE Consider the following:

- You cannot specify a new name for an entity. It's current name will be appended to the specified destination.
- When you move an entity, you are changing its location as shown in the desktop client, but not the physical location of the existing project directories or files on disk. However, this may determine the location of newly created files.

Options

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

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp **-pp***project_path*

Sets the current project for this command execution. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Sets the current project database for this command execution. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If it is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

-sp **-sp/@/***userID* | **Public**/*parent_workspace*/*child_workspace*

Specifies a public or private workspace. Note that user IDs are case-sensitive. To specify the Root Workspace, enter **/@/**RootWorkspace for the workspace value. If a workspace is not specified, the user's default workspace is used.

RootWorkspace for the workspace value. If a workspace is not specified, the user's default workspace is used.

Examples

Example 1: The following example shows two ways to move the versioned files ReadMe.txt and Resource.h from the project /bridge into the subproject /bridge/hlp.

```
pcli move -prC:\Users\All Users\Serena\VM\sampledb -idsusan -pp/bridge
    -dhlp ReadMe.txt Resource.h
pcli move -prC:\Users\All Users\Serena\VM\sampledb -idsusan -d/bridge/
    hlp /bridge/ReadMe.txt
    /bridge/Resource.h
```

Example 2: The following example moves the project /chess/client/board to the location /chess/board.

```
pcli move -prC:\Users\All Users\Serena\VM\sampledb -idsusan -d/chess /
    chess/client/board
```

Related Topics

For information about...	See...
Creating projects	"CreateProject command" on page 58
Creating project databases	"CreateProjectDB command" on page 61
Deleting Version Manager entities	"Delete command" on page 65
Importing Archives	"ImportArchives command" on page 92

PromoteGroup command

Use the PromoteGroup command to promote versioned files to the next promotion group.

Privileges required: Promotion Group

Alias: PG

Exit codes: Returns no exit code if successful.

-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax: **PromoteGroup -g***groupFrom* [**-yb** | **-nb**] [**-z**] *entity*

Where *groupfrom* specifies the name of the promotion group to promote from.

Options

-nb Does not allow promotion across branches.

-qr Ignores revisions, labels, and groups that are not located in the archives. Does not print warning messages or set a non-zero exit code.

-yb Allows promotion across branches.

-z Recursively promote versioned items in subprojects.

Related Topics

For information about...	See...
Assigning promotion groups	"AssignGroup command" on page 53
Changing to a different promotion group	"ChangeGroup command" on page 56
Deleting promotion groups	"DeleteGroup command" on page 67

Put command

Use the Put command to check in a revision of the specified versioned files. By default, files are checked in from the location to which they were checked out. If a file has no check out location (as when a file is locked, but not checked out), it will be checked in from its default workfile location as defined in the current workspace, unless you specify an alternate location.

If you do not enter a revision description using the `-m` option, you will be prompted for a description. To end the file description, place a period (.) on a line by itself.

Privileges required Check In Tip, and Check in Branch. Other privileges may also be required, such as Create Branch (for checking the first revision in to a branch) and Add Version Label (for using the **`-v`** option). For more information on privileges, see the Administrator's Guide.

Alias None.

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax `Put [options] [entity] ...`

Where entity specifies a versioned file, folder, project, or project database for which you want to check in a revision.

Options

-a **-a***path*

Specifies an alternate location from which to check in workfiles, rather than the location to which they were checked out. The check out location is used unless you specify either the **-a** or **-fw** option.

-b Ignore leading and trailing blank spaces when testing whether the revision you are checking in is unchanged.

-bp **-bp***path*

Specifies the base project path to use in calculating workfile locations when **-a** or **-o** has been specified. For multiple-file operations, *path* must be the entity path to a common parent of each of the items being checked in.

-f Force put even if workfile is unchanged.

Note, this option is depreciated. You should use the **-yf** option instead.

-fb Force the creation of a new branch.

-fv Specifies that the version label specified by the **-v** option is a floating label.

-fw Forces the use of the workspace that is currently in effect, overriding any recorded check out location. The check out location is used unless you specify either the **-a** or **-fw** option.

-g **-g***promotion_group*

Specifies whether or not a revision is being identified by its promotion group. This option functions in two different ways:

- **-g***promotion_group* identifies the promotion group to which the revision is currently assigned.
- **-g** indicates that one is not identifying the revision by promotion group.

Note, if there is only one locked revision, this option is ignored.

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

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-k Keep the original workfile unchanged.

-l **-l**

Performs a Get with lock after the Put operation.

-m Specifies the change description for the revision. This option can be used in two ways:

- **-m***description* specifies a description at the command-line.
- **-m@***file* obtains a description from a text file.

NOTE If you do not use the -m option, you will be prompted to enter a description. When prompted, enter the description, using the ENTER key to create new lines as desired. To end the description, place a period (.) on a line by itself.

-nf Aborts the Put operation if the workfile is unchanged or older.

-nv Prevents the version label specified by the **-v** option from being assigned to the new revision.

-o Overrides the workfile location defined in the project and bases workfile paths on the project structure.

-pp **-pp***project_path*

Sets the current project for this command execution. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Sets the current project database for this command execution. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If it is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

-r **-r***revision*

Specifies the revision number to use for the new revision.

NOTE This option results in a Yes/No prompt. Use the **run -y** command to confirm this action in your scripts.

-sp **-sp/@/***userID* | **Public**/*parent_workspace*/*child_workspace*

Specifies a public or private workspace. Note that user IDs are case-sensitive. To specify the Root Workspace, enter **/@/**RootWorkspace for the workspace value. If a workspace is not specified, the user's default workspace is used.

-u Updates workfile. Does a Get operation without a lock, after the Put operation is complete.

-v **-v***version*

Assigns a version label to the new revision. Note, you can use the **-yv** or **-nv** option to automatically answer Yes or No to prompts to reassign an existing label.

-yf Checks in the revision even if the workfile is unchanged or older.

-ym Uses the change description specified by the **-m** option for all versioned items.

-yv Reassigns the version label specified by the **-v** option to the new revision, if the version label exists.

-z Includes revisions in subprojects.

Examples **Example 1:** The following example checks in the versioned file server.bat in the project /chess/server and assigns the version label REL_1.0 to the new revision—even if that label is currently assigned to another revision. Also, the new revision is given the change description NT Support.

```
pcli Put -prC:\Users\All Users\Serena\VM\sampledb -idjohn -vREL_1.0 -yv -m"NT Support"
   /chess/server/server.bat
```

Example 2: The following example shows two ways to check in the versioned files server.bat and shutdown.txt in the project /chess/server from the alternate directory C:\test. The example uses the basepath (**-bp**) option to indicate that the directory C:\test is considered to be at the project hierarchy level of /chess/server. Thus, Put checks in the workfiles C:\test\server.bat and C:\test\shutdown.txt.

```
pcli Put -prC:\Users\All Users\Serena\VM\sampledb -idjohn -aC:\test -o -bp/chess/server
   -pp/chess/server server.bat shutdown.txt
pcli Put -prC:\Users\All Users\Serena\VM\sampledb -idjohn -aC:\test -o -bp/chess/server
   /chess/server/server.bat /chess/server/shutdown.txt
```

Example 3: The following example is the same as Example 2 except that the directory C:\test contains subdirectories matching the project structure from the project database root down. The example sets the basepath to / (root), so Put checks in the workfiles C:\test\chess\server\server.bat and C:\test\chess\server\shutdown.txt.

```
pcli Put -prC:\Users\All Users\Serena\VM\sampledb -idjohn -aC:\test -o -bp/
   -pp/chess/server server.bat shutdown.txt
pcli Put -prC:\Users\All Users\Serena\VM\sampledb -idjohn -aC:\test -o -bp/
   /chess/server/server.bat /chess/server/shutdown.txt
```

Related Topics

For information about...	See...
Listing the workfile location of projects and versioned files	"GetWorkLocation command" on page 88
Checking out revisions	"Get command" on page 79
Unlocking revisions	"Unlock command" on page 162
Using the Run command	"Run command" on page 139

Readline command

The Readline command enables you to write a script that prompts the user for input from the command line. This command reads a single line of input and places it in the specified variable.

Privileges required: None.

Alias: None.

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax

Readline [**-h**] {**-a***variable_name* | **-v***variable_name*}

Where *variable_name* is the name of the variable to hold the user response. Either the -a or -v option is required.

Options

-a **-a***variable_name*

Places each word of a line of user input into a different element of the named variable's array.

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

-v **-v***variable_name*

Places a line of user input into the first element of the named variable.

Example

The following example sets a variable named configfile to the name and path of a configuration file, then queries the user for input, and finally uses the If command to perform an action according to the user's response:

```
set -vconfigfile D:\producta\archives\cbo09asd.cfg
echo Do you want to change the configuration file for the \ project
    database to $configfile? y or n
readline -vanswer
if test $answer=y
{
   setconfigfile -prD:\producta -c${configfile}
}
else
{
```

```
    echo The configuration file was not changed.
}
```

Related Topics

For information about...	See...
Other commands used in scripts	ArraySize on page 52, Break on page 54, Calc on page 55, Continue on page 57, Echo on page 63, Exit on page 73, For on page 76, If on page 91, Return on page 138, Run on page 139, Set on page 142, Test on page 160, and While on page 175.

RenameLabel command

Use the RenameLabel to rename a version label in specified versioned files.

Privileges required: Modify Version Label, and Delete Version Label. For more information on privileges, see the Administrator's Guide.

Alias: RL

Exit codes:

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax: **RenameLabel** [*options*] **-vf***from_label* **-vt***to_label entity ...*

Where:

-vf*from_label* specifies the existing name of the version label.

-vt*to_label* specifies the new name of the version label.

entity specifies the folder, project database, project, or versioned file that contains the version label.

NOTE If the *to_label* already exists on a revision in a versioned file, it will, in effect, be reassigned to the revision that is associated with the *from_label*.

Options

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

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp **-pp***project_path*

Sets the current project for this command execution. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Sets the current project database for this command execution. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If it is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qr Ignores revisions, labels, and groups that are not located in the archives. Does not print warning messages or set a non-zero exit code.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

-sp **-sp/@/***userID* | **Public**/*parent_workspace*/*child_workspace*

Specifies a public or private workspace. Note that user IDs are case-sensitive. To specify the Root Workspace, enter **/@/**RootWorkspace for the workspace value. If a workspace is not specified, the user's default workspace is used.

-z Includes versioned files in subprojects.

Examples **Example 1:** The following example shows two ways to rename the label TEST_BUILD to REL_1.1 on the files server.bat and shutdown.txt in the /chess/server project.

```
pcli RenameLabel -prC:\Users\All Users\Serena\VM\sampledb -idjohn -vfTEST_BUILD
```

```
   -vtREL_1.1 -pp/chess/server server.bat shutdown.txt
pcli RenameLabel -prC:\Users\All Users\Serena\VM\sampledb -idjohn -vfTEST_BUILD
   -vtREL_1.1 /chess/server/server.bat /chess/server/shutdown.txt
```

Example 2: The following example renames the label TEST_BUILD to REL_1.1 for all versioned files in and below the chess project.

```
pcli RenameLabel -prC:\Users\All Users\Serena\VM\sampledb -idjohn -vfTEST_BUILD
   -vtREL_1.1 -z /chess
```

Related Topics

For information about...	See...
Deleting labels	"DeleteLabel command" on page 68
Listing floating labels	"ListFloatingLabels command" on page 114

Return command

Use the Return command to cause the current function to return with the specified status. The command returns from the next higher enclosing function.

NOTE To completely exit from a script, use the Exit command. For more information, see "Exit command" on page 73.

Privileges required: None.

Alias: None.

Exit codes: Returns no exit code when successful.

-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax: **Return** [*options*] [*status*]

Where *status* specifies the value to return, which can be any string (for example, True). If you do not specify status, a value of 0 (zero) is returned as the status.

Options

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

Related Topics

For information about...	See...
Other commands used in scripts	ArraySize on page 52, Break on page 54, Calc on page 55, Continue on page 57, Echo on page 63, Exit on page 73, For on page 76, If on page 91, Readline on page 135, Run on page 139, Set on page 142, Test on page 160, and While on page 175.

Run command

Execute PCLI commands or external programs

Use the Run command to execute PCLI commands or external programs. The PCLI commands may be in a script file, in a function that was previously defined, or a single command directly specified. See "PCLI Functions" on page 31 for information on defining functions.

Privileges required

None.

Alias

None.

Exit codes

Returns the status of the command to run when successful.

-2 PCLI command not found
-3 A non-PCLI related error or a command-specific error
-6 An invalid argument was specified
-7 An argument for a flag that is not needed
-8 A missing argument for a flag
-9 Wrong type was specified for an option's argument
-10 The specified file name cannot be read
-11 A required argument is missing
-12 A security exception occurred
-13 An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax

Run [**-h**] [**-b**] [**-ns**] [*I/O_args*] **-s***file_name* [*arg...*]

or
Run [**-h**] [**-b**] [**-ns**] [*I/O_args*] **-f***function_name* [*arg...*]

or
Run [**-h**] [**-b**] [**-ns**] [*I/O_args*] **-e** *external_command_line*

or
Run [**-h**] [**-b**] [**-ns**] [*I/O_args*] *pcli_command_line*

Where:

I/O_args are the redirection options: **->**, **->>**, **-xo**, **-xe**, **-ao**, **-ae**, **-vo**, **-va**, **-vi**.

file_name specifies a PCLI script file.

arg specifies the value or values that the script or function is to operate on. Separate multiple values with a blank space.

function_name specifies a PCLI function by name.

external_command_line specifies an external program.

pcli_command_line specifies a PCLI command.

Options

To substitute variables and use list files for additional options, see "Command Options" on page 16 for more information.

-> **->***file_name*

Redirects output and error messages to the specified file. Creates the file if it does not exist. If the file exists, overwrite it, unless the **->>** option is specified, then append the output to the existing file.

->> **->>***file_name*

Appends output and error messages to an existing file.

-ae **-ae***file_name*

Appends error messages to the specified file, if the file exists.

-ao **-ao***file_name*

Appends standard output to the specified file, if the file exists.

-b **-b**[*variable_name*]

Runs the command in the background. Optionally, specify a variable name in which to place the status when the command completes.

-e **-e** *program_name*

Executes an external program, such as an operating system command, in the native operating system shell. Note that you must enter a space after the **-e** option.

NOTE On Windows , when you execute internal native commands using the run **-e** command, it's a good idea to first start the command interpreter:

run -e cmd /c *internal_command_line*

Some examples of the internal native commands for Windows are: cd, chdir, cls, copy, del, dir, erase, md, mkdir, more, move, rd, ren, rename, rmdir, type, and xcopy.

-f **-f***function_name*

Executes a PCLI function by name.

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

-n Force No response to all Yes/No queries.

-ns Prevents quotation marks from being stripped from all arguments except those that are listed as options of the Run command itself. See Example 4 on page 27.

-q Quiet mode. Only report errors.

-s **-s***script_name*

Executes PCLI commands contained in a script file.

-tq Trims the quotes from the start and end of an argument. Use this option to expand variables from double quoted strings and preserve any double quotes in the string's value. For example:

```
set VAR='This variable has "double quotes" in its value'
set -tq VAR_COPY="${VAR}"
echo -tq "VAR_COPY is set to: ${VAR_COPY}."
```

This example prints:

```
VAR_COPY is set to: This variable has "double quotes" in its value.
```

-va **-va***variable_name*

Redirects standard output to the specified array variable. This option places each line of output in a different element of the named variable array starting at the specified integer index. If the index is not specified, starts at zero. Use the **-vi** option to set an integer index.

-vi **-vi***index*

Sets the integer to use to start numbering the array variable. This option works in conjunction with the `-va` option.

-vo **-vo***variable_name*

Redirects standard output to the specified variable.

-xe **-xe***file_name*

Redirects error messages to the specified file, if the file exists.

-xo **-xo***file_name*

Redirects standard output to the specified file, if the file exists.

-y Force Yes response to all Yes/No queries.

Examples **Example 1:** The following example executes a PCLI script named checkin.pcli:

```
pcli run -scheckin.pcli
```

Example 2: The following example passes two arguments to a PCLI script:

```
pcli run -sscript.pcli D:\producta /proj1
```

And, the script looks like this:

```
# Set the first argument passed to the variable PDB
# The variable is quoted in case the value passed
# contains spaces
set -vPDB "$1"
# Set the second argument passed to the variable PROJECT
set -vPROJECT "$2"
# Use the values of the variables in the command line
```

```
getconfigfile -pr"$PDB" -pp"$PROJECT"
```

Example 3: The following example runs the DOS command Del to delete the files.tmp file on Windows:

```
run -e cmd /c del files.tmp
```

Example 4: The following example redirects the output of the ListVersionedFiles command into a file.

```
run ->files.tmp listversionedfiles -prD:\producta /bridge
```

Example 5: The following example allows you to run a CLI command on a File Server path map that is under SSO authentication (CLI commands do not support the SSO/CAC login source). In order to provide the File Server with valid SSO/CAC credentials, CLI commands can be executed via the **run -e** command in a PCLI script. Note, the script must first execute a PCLI command against the path map.

```
# Define PDB
set -vPCLI_PR "//vmfs/sso_pdb"
# Define UserID:Password
set -vPCLI_ID "UserID:Password"

# Run a dummy PCLI command to authenticate to the PDB, which will get
    the SSO token
set -vUserID $[WhoAmI]
if [ "$UserID" = "" ]
{
        echo -ns User "$PCLI_ID" not valid for PDB "$PCLI_PR".
        exit 1
}

# Run CLI command by way of "run -e", which will pass the token obtained
    by the previous PCLI command

run -ns -e YourCLICommandHere
```

Related Topics

For information about...	See...
Other commands used in scripts	ArraySize on page 52, Break on page 54, Calc on page 55, Continue on page 57, Echo on page 63, Exit on page 73, For on page 76, If on page 91, Readline on page 135, Return on page 138, Set on page 142, Test on page 160, and While on page 175.

Set command

Set variable assignments

Use the Set command to assign a value to a PCLI variable. For information about PCLI variables, see "Variables" on page 18.

PCLI variables are automatically defined when first set.

When setting a variable to a value that contains spaces, you must enclose the value in quotation marks. For example:

```
set -vPCLI_PR "D:\My Apps\producta"
```

If you are setting a variable to the value of another variable that contains spaces, you must enclose the variable in quotation marks and escape the quotation marks so that the quotation marks are considered literal characters. For example, **set -v**PCLI_PR \"$1\".

If the value that you are assigning to the variable contains a hyphen followed by a letter, you must enclose the string in quotation marks. For example, "-c" or "-c$i". This is necessary so that the command does not consider the string an option in the command line.

Privileges required: None.

Alias:
pr equals **Set -v**PCLI_PR
pp equals **Set -v**PCLI_PP
sp equals **Set -v**PCLI_SP
local equals **Set -l**

Exit codes:

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax 1
Set [**-h**] [**-l**] [**-r**] **-v***variable_name* [**-i***index*] {*text* | *$variable* | $(*command*) | @*list_file*}...

Or
Set [**-h**] [**-l**] [**-r**] [**-i***index*] *variable_name*[*[index]*]={*text* | *$variable* | $(*command*) | @*list_file*}...

Syntax 2
Set [**-h**] [**-l**] [**-r**] **-a -v***variable_name* [**-i***index*] {*text* | *$variable* | $(*command*) | @*list_file*}...

Or
Set [**-h**] [**-l**] [**-r**] **-a** [**-i***index*] *variable_name*[*[index]*]={*text* | *$variable* | $(*command*) | @*list_file*}...

Syntax 3
Set [**-h**] [**-l**] [**-r**] **-v***variable_name* **-i***index* {*text* | *$variable* | $(*command*) | @*list_file*}...

Syntax 4
Set [**-h**] [**-l**] [**-r**] **-a***variable_name* **-v***variable_name2* [**-i***integer*]

Where *variable_name* specifies the name of the variable to set.

For an explanation of $(*command*), see "Command Output" on page 21. For an explanation of @*list_file*, see "List Files" on page 21.

Syntax 1 enables you to define a typical string variable. This form of variable assignment places text, the value of a variable, the output of a command, or the contents of a list file into a single element of the named variable. If no index is specified, all text is placed in the first element of the variable, which is 0.

Syntax 2 enables you to define a sequential array. This form does array assignment, where each word of the source is placed in a different element in the target variable array. Word splitting is affected by the PCLI_IFS variable (see "Word Splitting" on page 22). If no index is specified, the first word is placed in the first element of the variable array, which is 0. In the case of a sequential array, the index must be an integer.

Syntax 3 enables you to define an associative array. In the case of an associative array, the **-i** option is required with any character string as the index. See Example 4 on page 145.

Syntax 4 does array assignment, where the entire array of the variable specified by the **-a** option is copied element by element into the array of the second named variable specified by the **-v** option. In this case, the values for the variable are copied from the array of the variable specified by the **-a** option instead of copied from text, command output, or a list file as in the second form of the command.

Options

To substitute variables and use list files for additional options, see "Command Options" on page 16 for more information.

-a **-a**[*variable_name*]

Sequentially assigns array values at the starting index. The specified index must be an integer (**-i** option). If *variable_name* is specified (as in the third form of the syntax), then this is the name of a variable array to copy from. If no *variable_name* is specified (as in the second form of the syntax), then the values for the array are obtained from the text, $variable, $(command), or @list_file specified.

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

-i **-i***index*

Sets the array index of the variable. If the **-a** flag is used, this must be an integer; otherwise, the index may be any string.

-l Makes the variable a local variable.

-ns Prevents quotation marks from being stripped from all arguments except those that are listed as options of the Set command itself.

-r This option applies only to setting the PCLI_IFS variable. Allows you to set the following unprintable characters for the PCLI_IFS variable:

\n = newline
\r = return
\t = tab
\f = form feed
\b = backspace

You must place quotation marks around the characters when setting them, for example:
Set -r -vPCLI_IFS "\n\b"

Or
```
Set -r PCLI_IFS="\n\b"
```

-tq Trims the quotes from the start and end of an argument. Use this option to expand variables from double quoted strings and preserve any double quotes in the string's value. For example:

```
set VAR='This variable has "double quotes" in its value'
set -tq VAR_COPY="${VAR}"
echo -tq "VAR_COPY is set to: ${VAR_COPY}."
```

This example prints:

```
VAR_COPY is set to: This variable has "double quotes" in its value.
```

-v **-v***variable_name*

Names the variable to be set.

Examples **Example 1:** The following example sets the PCLI_PR variable to the BoardGames project database:

```
set -vPCLI_PR H:\BoardGames
```

Or
```
set PCLI_PR=H:\BoardGames
```

Example 2: The following example sets a variable array to the items output from a command execution.

```
set -a -vfiles $[listversionedfiles -prD:\producta /proj1]
```

Or
```
set -a files=$[listversionedfiles -prD:\producta /proj1]
```

Example 3: The following example sets a variable named test, which is defined with an array. Then, the items from the test variable array are copied to the array of a second variable named test2.

```
set -a -vtest item1 item2 item3
set -atest -vtest2
```

Example 4: The following is a simple example of how to use associative arrays to provide an easy way to determine which scripts or functions to run based on the name of the test.

```
# Execute this script with a first argument of the desired
# test name (That_test, This_test, Other_test) and then
# whatever arguments are appropriate for that test. Arguments
# not specified will be null.
set -vfuncname $1
set  -vtestfunc -iThis_test 'ThisTestFunc'
set  -vtestfunc -iThat_test 'ThatTestFunc'
if [ "$testfunc[$funcname]" = "" ]
{
   echo Unrecognized test name
   echo Valid test names are:
   echo "   This_test"
   echo "   That_test"
   exit
}
```

```
ThisTestFunc()
{
   # This function does some important thing
   echo arg1=$1
   echo arg2=$2
   echo arg3=$3
}
ThatTestFunc()
{
   # This function does another important thing
   echo arg1=$1
   echo arg2=$2
   echo arg3=$3
   echo arg4=$4
}
echo $funcname->$testfunc[$funcname]
run -f$testfunc[$funcname] $2 $3 $4
```

Example 5: The following is an example of how to define an array variable and then indicate whether you are modifying the existing array or overwriting the array with new values. This is useful when you combine arrays or modify certain elements of an array (such as adding quotation marks around certain values). The following example modifies the value of the FOO variable.

```
set -a -vFOO First time set
echo $FOO[]
set -a -vFOO New values
echo $FOO
set -a -vFOO -il and improved
echo $FOO[]
```

Output:

```
First time set
New values
New and improved
```

In this example, the first two Set commands simply set the FOO variable to the values specified. The third Set command uses the **-i** option to indicate that the variable FOO is to be modified, not just set to a new value. The **-i**`l` option tells PCLI to modify the values of the array's elements starting with `l`.

Related Topics

For information about...	See...
Displaying the values of PCLI variables	"Echo command" on page 71
Other commands used in scripts	ArraySize on page 52, Break on page 54, Calc on page 55, Continue on page 57, Exit on page 73, For on page 76, If on page 91, Readline on page 135, Return on page 138, Test on page 160, and While on page 175.

SetArchiveInfo command

Use the SetArchiveInfo command to modify the archive attribute information for the specified versioned files.

Privileges required: Modify Archive Properties. For more information on privileges, see the Administrator's Guide.

Alias: SAI

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax: **SetArchiveInfo** [*options*] [*entity*] ...

Where *entity* specifies a folder, project database, project, or versioned file.

Options

-a **-a***ID_list*

Adds to or deletes an access list. The **-a** option appends the user IDs specified in a comma-separated list, *ID_list*, to any existing access list in the archive or creates an access list if one doesn't already exists. To delete an access list, use the **-a** option without specifying an *ID_list*.

-cc Disables compression of workfile images.

-cd Disables compression of deltas.

-ce Disables exclusive locking.

-cg Disables delta generation.

-ck Disables keyword expansion.

-cl Disables lock checking.

-ct Disables file translation.

-cw Disables archive write protection.

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

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp **-pp***project_path*

Sets the current project for this command execution. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Sets the current project database for this command execution. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If it is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

-sc Enables compression of workfile images.

-sd Enables compression of deltas.

-se Enables exclusive locking.

-sg Enables delta generation.

-sk Enables keyword expansion.

-sl Enables lock checking.

-sp **-sp/@/***userID* | **Public**/*parent_workspace*/*child_workspace*

Specifies a public or private workspace. Note that user IDs are case-sensitive. To specify the Root Workspace, enter **/@/**RootWorkspace for the workspace value. If a workspace is not specified, the user's default workspace is used.

-st Enables file translation.

-sw Enables archive write protection.

-t Sets the workfile description. This option can be used in two ways:

- **-t***description* specifies a description at the command-line.
- **-t@***file* obtains a description from a text file.

-w **-w***workfile*

Sets the workfile name stored in the archive. This value is used by all Command-Line Interface (CLI) commands and to determine the name of the file placed in a Reference Directory (if any) during check in.

IMPORTANT! Project Command-Line Interface (PCLI) commands and the Version Manager desktop client ***do not*** use the workfile name stored in the archive--***except*** to determine the name of the file placed in a Reference Directory (if any) during check in. For all other operations the name of the versioned file itself is used.

-z Includes revisions in subprojects.

Examples **Example 1:** The following example clears the WRITEPROTECT attribute on all archives associated with the versioned files at or below the /class project.

```
pcli SetArchiveInfo -prC:\Users\All Users\Serena\VM\sampledb -idsusan -cw -z /chess
```

Example 2: The following example changes the workfile description of all versioned files in the /chess/client/images project to *Game images*.

```
pcli SetArchiveInfo -prC:\Users\All Users\Serena\VM\sampledb -idsusan -t"Game images"
   /chess/client/images
```

Related Topics

For information about...	See...
Listing versioned files	"ListVersionedFiles command" on page 122
Reporting archive and revision information for versioned files	"Vlog command" on page 168

SetArchiveLocation command

Set the archive location for the specified entity

Use the SetArchiveLocation command to set the archive location for the specified project database, project, or versioned file. Typically, this location is on a networked file system, a location to which all authorized users have access. By default, when you create a project, the archive location is a directory beneath the project database location, but you can specify a different archive location using this command.

NOTE This command is not functional for Version Manager 5.3/6.0 project roots.

Privileges required Modify Project. For more information on privileges, see the Administrator's Guide.

Alias SAL

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error

-6 An invalid argument was specified
-7 An argument for a flag that is not needed
-8 A missing argument for a flag
-9 Wrong type was specified for an option's argument
-10 The specified file name cannot be read
-11 A required argument is missing
-12 A security exception occurred
-13 An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax **SetArchiveLocation** [*options*] **-a***archive_location* [*entity*]

Where:

archive_location specifies the path to the archive directory. You can set the archive location of a project to either a relative path, for example, `/prj1`, or a full absolute path, for example, `H:\Samples\archives\prj1`. If you set the path to a relative path, the location will be appended to the parent's computed archive location. The **-a** option is required.

entity specifies a project or versioned file for which you are setting the archive location. You need not specify an entity if you want to set the archive location for the current project or project database.

NOTE The current project is set using either the **-pp** option (as described below) or the PCLI_PP variable. A project database is set by using either the **-pr** option (as described below) or the PCLI_PR variable.

Options

To substitute variables and use list files for additional options, see "Command Options" on page 16 for more information.

-a **-a***archive_location*

Specifies the path to the archive directory. You can set the archive location of a project to either a relative path, for example, `/prj1`, or a full absolute path, for example, `H:\Samples\archives\prj1`. If you set the path to a relative path, the location will be appended to the parent's computed archive location. This is required.

-h Display help for the command. The action terminates after it processes the **-h** option even if you specify other options.

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp **-pp***project_path*

Specifies the current project or folder for which to set the archive location. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Specifies the project database for which to set an archive location. This option overrides the value of the PCLI_PR variable for a single command execution. A project database is required for this command. If no project database is specified, the PLCI_PR variable is used. If it is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

Examples **Example 1:** The following example changes the archive location for the Checkers project to the specified directory.

```
pcli setarchivelocation -prD:\producta -pp/checkers -aH:\archives\producta\checkers
```

Example 2: The following example sets the archive location for a project database to a value specified in a variable named archivelocation:

```
pcli setarchivelocation -prD:\productb -a$archivelocation
```

Related Topics

For information about...	See...
Getting the archive location	"GetArchiveLocation command" on page 84

SetConfigFile command

Use the SetConfigFile command to set the configuration file for a project database or project/folder.

Privileges required Modify Project. For more information on privileges, see the Administrator's Guide.

Alias SCF

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed

-8 A missing argument for a flag
-9 Wrong type was specified for an option's argument
-10 The specified file name cannot be read
-11 A required argument is missing
-12 A security exception occurred
-13 An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax **SetConfigFile** [*options*] **-c***configfile_location* [*entity*]

Where:

configfile_location specifies the path to and name of the configuration file. If the **-c** option is not specified, any existing reference to a configuration file is removed.

entity specifies a project/folder for which you are setting the configuration file. You need not specify an entity if you want to set the configuration file for the current project/folder or project database.

NOTE The current project/folder is set using either the **-pp** option (as described below) or the PCLI_PP variable. A project database is set by using either the **-pr** option (as described below) or the PCLI_PR variable.

Options

To substitute variables and use list files for additional options, see "Command Options" on page 16 for more information.

-c **-c***configfile_location*

Specifies the path to and the name of the configuration file. If the **-c** option is not specified, any existing reference to a configuration file is removed.

-h Display help for the command. The action terminates after it processes the **-h** option even if you specify other options.

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id**:*PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp **-pp***project_path*

Specifies the project or folder for which you want to set the configuration file. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Specifies the project database for which you want to set the configuration file. This option overrides the value of the PCLI_PR variable for a single command execution. A project database is required for this command. If no project database is specified, the PLCI_PR variable is used.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

Examples **Example 1:** The following example sets the checkers.cfg file for the Checkers project in the project database located in H:\BoardGames.

```
pcli setconfigfile -prH:\BoardGames -pp/checkers -cH:\archives\dev\checkers\checkers.cfg
```

Example 2: The following example sets the configuration file for the project database located in H:\V60Planet to a file specified in a list file named config.txt. Note that in this example, the project database requires a login (uses the **-id** option).

```
pcli setconfigfile -prH:\V60Planet -c@config.txt -idAnnaB
```

Related Topics

For information about...	See...
Getting the location of the configuration file	"GetConfigFile command" on page 86

SetDefaultWorkspace

Use the SetDefaultWorkspace command to set the default workspace for a user or to display your current default workspace for the specified project database or project.

Privileges required SuperUser when setting another user's default workspace (**-u** option). No privileges are required when setting or displaying your own default workspace. For more information on privileges, see the Administrator's Guide.

Alias SDW

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing

-12 A security exception occurred
-13 An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax **SetDefaultWorkspace** [*options*]

NOTE A project database is set by using either the **-pr** option (as described below) or the PCLI_PR variable.

Options

To substitute variables and use list files for additional options, see "Command Options" on page 16 for more information.

-h Display help for the command. The action terminates after it processes the **-h** option even if you specify other options.

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id**:*PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pr **-pr***project_database*

Specifies the project database for which you want to set the workfile location. This option overrides the PCLI_PR variable for a single command execution. You must specify a project database for this command. If no project database is specified, the PLCI_PR variable is used. If it is not defined, then an error message is displayed and the command is aborted.

-sp **-sp***workspace*

Specifies the default workspace for the target user. If omitted, the default workspace for the **-id** user is used.

-u **-u***userID*

Specifies the target user. This option may be specified only if the **-id** user is a SuperUser. If omitted, the **-id** user is the target user.

Examples

Example 1: The following example shows the user (john_doe) specifying a private workspace (John) to use by default for the BookPDB project database.

```
SetDefaultWorkspace -prH:\pdbs\BookPDB -sp/@/john_doe/John
```

Example 2: The following example shows the administrator specifying a public workspace (Doc) to use by default when the user (john_doe) accesses the BookPDB project database.

```
SetDefaultWorkspace -idAdmin:Admin -prH:\pdbs\BookPDB -sp/@/Public/Doc
    -u/john_doe
```

Example 3: The following example displays the default workspace currently in effect for the user issuing the command in the BookPDB project database.

```
SetDefaultWorkspace -prH:\pdbs\BookPDB
```

SetWorkLocation command

Use the SetWorkLocation command to set the workfile location for a project database, project/folder, or versioned file. You must specify either an absolute path for the workfile location, or a path relative to the parent project's workfile location. If you specify a relative path, the location specified will be appended to the parent's archive location.

If the selected project's workfile location is currently absolute, you cannot change the workfile location to make it relative. You must specify an absolute workfile location.

NOTE Workfile location is a workspace-sensitive attribute in Version Manager. When you set a workfile location, the location is stored in a user's default workspace by default. If you want to store the workfile location in a workspace other than the user's default, you must specify which workspace by using the **-sp** option (described in the "Options" section below).

Privileges required

Modify Project. For more information on privileges, see the Administrator's Guide.

Alias

SWL

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax

SetWorkLocation [*options*] **-w***work_location* [*entity*]

Where:

work_location specifies the path to the workfile location, a file system path to a directory where files are checked out. The **-w** option is required.

entity specifies the project/folder or versioned file for which you are setting a workfile location. You need not specify an entity if you want to set the workfile location for the current project/folder or project database.

NOTE The current project/folder is set using either the **-pp** option (as described below) or the PCLI_PP variable. A project database is set by using either the **-pr** option (as described below) or the PCLI_PR variable.

Options

To substitute variables and use list files for additional options, see "Command Options" on page 16 for more information.

-h Display help for the command. The action terminates after it processes the **-h** option even if you specify other options.

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp **-pp***project_path*

Specifies the project or folder for which you want to set the workfile location. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Specifies the project database for which you want to set the workfile location. This option overrides the PCLI_PR variable for a single command execution. You must specify a project database for this command. If no project database is specified, the PLCI_PR variable is used. If it is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

-sp **-sp**[**/@**/*userID*] | [**/@**/*Public*]/*parent_workspace*[/*child_workspace*]

Specifies the active workspace (which will be the target of this command). It can be a public or private workspace. Note that user IDs are case-sensitive. To specify the Root Workspace, enter **/@**/RootWorkspace for the workspace value. If a workspace is not specified, the user's default workspace is used.

The specified workspace must already exist. Use the CreateWorkspace command to create a new workspace.

-w **-w***work_location*

Specifies the path to the workfile location, a file system path to a directory where files are checked out. This is required.

Examples **Example 1:** The following example changes the workfile location for the Checkers project to the location specified. The workfile location is set in the user's default workspace because no **-sp** option is specified.

The period (.) at the end of the example specifies that the entity is the current project (the current project is specified with the **-pp** option). Specifying an entity is optional for this command, but since other PCLI commands require an entity, it is good coding practice to always specify one.

```
pcli setworklocation -prH:\Boardgames -pp/checkers -wD:\MyWork\checkers .
```

Example 2: The following example sets the workfile location for a project database that requires a login. The forward slash at the end of the example ensures that root entity will be used even if the PCLI_PP environment variable has been set.

```
pcli setworklocation -pr"C:\Users\All Users\Serena\VM\SampleDB" -idAdmin
   -wD:\MyWork\checkers /
```

Example 3: The following example sets the workfile location of Project1 to a workfile location on a local drive for the specified workspace. The workfile location is set in the private workspace of user ID, mikaylap, named development.

```
pcli setworklocation -prH:\Samples -sp/@/mikaylap/development -wD:\MyWork\Prj1 /Project1
```

Example 4: The following example sets the workfile location of the file server.bat in the project /chess/server to that of the project by assigning it the value ".". During subsequent check out operations, this file will use the project's workfile location.

```
pcli setworklocation -prH:\Samples -w. /chess/server/server.bat
```

Related Topics

For information about...	See...
Getting the workfile location	"GetWorkLocation command" on page 88

SetWorkspaceAttributes

Use the SetWorkspaceAttributes command to set the default version, base version, branch version, and default lowest level promotion group for the current workspace. You can also display the values in effect for the specified project and workspace, including values inherited from higher level projects/workspaces.

Privileges required

Configure Project, if the workspace is public. No privileges are required to configure your own private workspaces. For more information on privileges, see the Administrator's Guide.

Alias

SWA

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax

SetWorkspaceAttributes [*options*] [*entity*]

Where *entity* specifies the project/folder for which you are setting workspace attributes. You need not specify an entity if you want to set the workspace attributes for the current project/folder or project database.

NOTE The current project/folder is set using either the **-pp** option (as described below) or the PCLI_PP variable. A project database is set by using either the **-pr** option (as described below) or the PCLI_PR variable.

Options

To substitute variables and use list files for additional options, see "Command Options" on page 16 for more information.

-ba **-ba***base_version*

Specifies the base version for the target workspace. To clear the base version in effect for the target workspace (to inherit one from a higher workspace or project entity), use this option with no value specified.

-br **-br***branch_version*

Specifies the branch version for the target workspace. To clear the branch version in effect for the target workspace (to inherit one from a higher workspace or project entity), use this option with no value specified.

-d **-d***default_version*

Specifies the default version for the target workspace. To clear the default version in effect for the target workspace (to inherit one from a higher workspace or project entity), use this option with no value specified.

-g **-g***promotion_group*

Specifies the default promotion group for the target workspace. To clear the default promotion group in effect for the target workspace (to inherit one from a higher workspace or project entity), use this option with no value specified.

-h Display help for the command. The action terminates after it processes the **-h** option even if you specify other options.

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp **-pp***project_path*

Specifies the project or folder for which you want to set the workfile location. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Specifies the project database for which you want to set the workfile location. This option overrides the PCLI_PR variable for a single command execution. You must specify a project database for this command. If no project database is specified, the PLCI_PR variable is used. If it is not defined, then an error message is displayed and the command is aborted.

-sp **-sp**[**/@**/*userID*] | [**/@**/*Public*]/*parent_workspace*[/*child_workspace*]

Specifies the active workspace (which will be the target of this command). It can be a public or private workspace. Note that user IDs are case-sensitive. To specify the Root Workspace, enter **/@**/RootWorkspace for the workspace value. If a workspace is not specified, the user's default workspace is used.

The specified workspace must already exist. Use the CreateWorkspace command to create a new workspace.

Examples

Example 1: The following example specifies a base (BASE_REV_8) and branch (REV_8) version in the Root Workspace for the / project.

```
SetWorkspaceAttributes -prH:\pdbs\DevPDB -sp/@/RootWorkspace
    -baBASE_REV_8 -brREV_8 /
```

Example 2: The following example specifies the default lowest level promotion group (iddAuthor) in the Doc workspace for the /tech project.

```
SetWorkspaceAttributes -prH:\pdbs\BookPDB -sp/@/Public/Doc
    -g/iddAuthor /tech
```

Example 3: The following example clears the default lowest level promotion group from the Doc workspace for the /tech project. The project will now inherit a default promotion group from a higher level workspace/entity. Note that the **-g** option is followed by two quote characters with no characters or spaces between them, an empty string.

```
SetWorkspaceAttributes -prH:\pdbs\BookPDB -sp/@/Public/Doc
    -g"" /tech
```

Test command

Use Test to repeat command in a loop

Use the Test command as a boolean expression evaluator for the If and While commands. Test returns 0 if true and 1 if false. No other output is generated. Test supports both alpha and numeric boolean-valued expressions, but does not perform numeric calculation (see "Calc command" on page 55).

Multipart expressions are allowed if they are connected by & or |. Parentheses and nested parentheses can be used to supersede the normal rules of precedence. The precedence rule is that & has precedence over | , and the other comparison operators have precedence over either & or |. For example:

```
If test $Foo = $Bar
If test $Foo = $Bar & $Doo >= $Dar
```

The following table defines the supported operators.

Operators	Description
A = B	A equals B.
A != B	A does not equal B.
A < B	A less than B.
A > B	A greater than B.
A >= B	A greater than or equals B.
A <= B	A less than or equals B.
&	The AND operator connects the above comparisons to form more complex expressions. This operator has precedence over the OR operator, below.
| (vertical bar)	The OR operator connects the above comparisons to form more complex expressions.

When the = or != operator is used in testing the equality or inequality of strings, wildcard characters are allowed in the first or second operand but not both. See the table on page 15 for a list of valid wildcard characters. Also, you can make the comparison of strings case-insensitive by using the **-i** option.

To compare a variable with an empty string, you must quote the variable and enter "" for the empty string, for example:

```
test "$foo" = ""
```

Privileges required: None.

Alias: None.

Exit codes

0	True
1	False
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax: **Test** [*options*] *expression*

Where *expression* supports both alpha and numeric boolean-valued expressions.

Options

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

-i Ignores the case of strings being compared when you use either the = or != operator. By default, case is considered during string comparisons.

Example: The following example sets the Configfile variable to the output of the GetConfigfile command execution. Then, the script looks at the value of the variable and compares it to a string (a configuration file path and name) using the Test command. If the value of the variable matches the string, the If statement is exited; if not, the SetConfigFile command is executed.

```
set -vConfigfile $(getconfigfile -prD:\producta)
If Test $Configfile=H:\producta\archives\cb09zjk.cfg
{
   exit
}
else
{
   setconfigfile -prD:\producta -cH:\producta\archives\cb09zjk.cfg
```

```
}
```

NOTE The Test command can also take the following form:

```
If [ $Configfile=H:\producta\archives\cb09zjk.cfg ]
```

Where brackets replace the word Test. Note that a space separates each bracket from the expression.

Special Considerations

Errors are produced by expressions with incorrect syntax. Examples are illegal characters in an expression and illegal combinations of operators.

Related Topics

For information about...	See...
Setting variables	"Set command" on page 142
Other commands used in scripts	ArraySize on page 52, Break on page 54, Calc on page 55, Continue on page 57, Echo on page 63, Exit on page 73, For on page 76, If on page 91, Readline on page 135, Return on page 138, and While on page 175.

Unlock command

Use the Unlock command to unlock a revision of the specified versioned files.

Privileges required

Unlock, or Break Lock to remove a lock owned by another user. For more information on privileges, see the Administrator's Guide.

Alias

None.

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax

Unlock [*options*] [*entity*] ...

Where entity specifies a folder, project database, project, or versioned file.

Options

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

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp **-pp***project_path*

Sets the current project for this command execution. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Sets the current project database for this command execution. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If it is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

-r **-r***revision*

Specifies the revision, promotion group, or version to act upon.

-sp **-sp/@/***userID* | **Public**/*parent_workspace*/*child_workspace*

Specifies a public or private workspace. Note that user IDs are case-sensitive. To specify the Root Workspace, enter **/@/**RootWorkspace for the workspace value. If a workspace is not specified, the user's default workspace is used.

-u Removes locks placed by specific users, as specified below:

- **-u** removes all locks by the current user.
- **-u***user* removes all locks by the specified user.
- **-u*** removes all locks by all users.

-z Includes revisions in subprojects.

Examples

Example 1: The following example removes all locks owned by `susan` from the project database, but only if you are logged in as `susan`.

```
pcli Unlock -prF:\Serena\vm\common\sampldb -idsusan -u -z /
```

Example 2: The following example removes all locks owned by `susan` regardless of the user you are logged in as, as long as your user ID (in this example, `admin`) has the required privilege (Break Lock).

```
pcli Unlock -prC:\Users\All Users\Serena\VM\sampledb -idadmin:secret -ususan -z /
```

Example 3: The following example removes all locks owned by all users, as long as your user ID (in this example, `admin`) has the required privilege (Break Lock).

```
pcli Unlock -prC:\Users\All Users\Serena\VM\sampledb -idadmin:secret -
    u* -z /
```

Related Topics

For information about...	See...
Getting revisions	"Get command" on page 79
Locking revisions	"Lock command" on page 126
Checking in revisions	"Put command" on page 131

UpdateProjectFolder command

Performs 5.3/6.0 desktop client menu action of updating archive

Use the UpdateProjectFolder command to reflect changes in the list of project files included in the archive directories (such as added or removed files). This command applies only to Version Manager 5.3/6.0 and earlier projects, and performs the Version Manager 5.3/6.0 desktop client menu command of Folder | Update Project Folder.

Privileges required

Must not have NoFolderUpdateProjectFolder. For more information on privileges, see the Administrator's Guide.

Alias

`UPF`

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax

UpdateProjectFolder [*options*] *project_name*

Where *project_name* specifies a 5.3/6.0 project or folder. You need not specify an entity if you want to update the current project or folder.

NOTE The current project/folder is set using either the **-pp** option (as described below) or the PCLI_PP variable.

Options

To substitute variables and use list files for additional options, see "Command Options" on page 16 for more information.

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

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id**:*PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp **-pp***project_path*

Specifies the project or folder to update. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Specifies the project root that contains the project you want to update. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If no project database is specified, the PLCI_PR variable is used. If it is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

Example The following example updates the project Excel in the project root located in H:\vm60planet. The project to operate on is specified in the entity argument rather than by using the **-pp** option.

```
pcli UpdateProjectFolder -prH:\vm60planet /excel
```

Related Topics

For information about...	See...
Listing active project databases	"ListProjectDB command" on page 116
Listing entities you specify	"List command" on page 102

Vdel command

Use the Vdel command to delete a revision of the specified versioned files.

Privileges required Delete Tip, and Delete Non-Tip. For more information on privileges, see the Administrator's Guide.

Alias None.

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax **Vdel** [*options*] [*entity*] ...

Where *entity* specifies a folder, project, project database, or versioned file.

CAUTION! This command permanently deletes revisions. Before using it, consider:

- If you specify a project as the entity, recursive behavior is implied.
- Any revisions you delete from an archive cannot be recovered except by replacing the affected archive files with backups. But restoring a revision from a backup will restore ***all*** revisions in the archive--any newer revisions will be lost.

Options

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

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp **-pp***project_path*

Sets the current project for this command execution. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Sets the current project database for this command execution. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If it is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

-r Deletes a specified revision or range of revisions, depending on usage:

- **-r**

  Deletes the default revision. If no default revision is defined, it deletes the latest trunk revision.

- **-r***revision*

  Deletes the specified revision.

- **-r***from_rev***to_rev*

  Deletes all revisions between *from_rev* and *to_rev* (inclusive). If *from_rev* is omitted, the first revision at the bottom of the trunk or branch indicated by *to_rev* is assumed. If *to_rev* is omitted, the tip revision on the tip of the trunk or branch indicated by *from_rev* is assumed.

- **-r**{[*revision*] | [*from_rev***to_rev*]}+

  Deletes all branches from the specified revisions.

-sp **-sp/@/***userID* | **Public/***parent_workspace/child_workspace*

Specifies a public or private workspace. Note that user IDs are case-sensitive. To specify the Root Workspace, enter **/@/**RootWorkspace for the workspace value. If a workspace is not specified, the user's default workspace is used.

-v Deletes revisions specified by a version label or range of version labels, depending on usage:

- **-v***version*

  Deletes the revisions specified by the version label.

- **-v***from_ver****** *to_ver*

  Deletes all revisions between the revisions identified by the version labels *from_ver* and *to_ver* (inclusive). If *from_ver* is omitted, the first revision at the bottom of the trunk or branch indicated by *to_ver* is assumed. If *to_ver* is omitted, the tip revision on the tip of the trunk or branch indicated by *from_ver* is assumed.

- **-v**{[*version*] | [*from_ver****** *to_ver*]}+

  Deletes all branches from the specified revisions.

Examples **Example 1:** The following example deletes revision 1.0 from the versioned file bridge.h.

```
pcli Vdel -prC:\Users\All Users\Serena\VM\sampledb -idjohn -r1.0 /bridge/bridge.h
```

Example 2: The following example deletes the revisions identified by the version label REL_1.0 from all files in and below the /chess project. Use **Run -y** to automatically answer Yes to all prompts.

```
pcli run -y Vdel -prC:\Users\All Users\Serena\VM\sampledb -idjohn -vREL_1.0 /chess
```

Example 3: The following example deletes all the revisions from the bottom of the trunk up to and including the revision specified by the version label REL_1.0, including any branches, from all files in and below the /chess project. Use **Run -y** to automatically answer Yes to all prompts.

```
pcli run -y Vdel -prC:\Users\All Users\Serena\VM\sampledb -idjohn -v*REL_1.0+ /chess
```

Related Topics

For information about...	See...
Deleting Version Manager entities	"Delete command" on page 65
Automatically answering prompts	"Run command" on page 139

Vlog command

Use the Vlog command to report archive and revision information about the specified versioned files.

Privileges required View Archive Header, and View Archive Revisions. For more information on privileges, see the Administrator's Guide.

Alias None.

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax **Vlog** [*options*] [*entity*] ...

Where *entity* specifies a folder, project, project database, or versioned file.

Options

-a **-a***user_id*[,*user_id*] ...

Limits report to revisions made by specified users. Note, separate multiple user IDs with commas.

-b Brief archive information only. Does not report revision information.

-bc **-bc**[*revision*]

Brief report of archives with a latest (tip) revision different from that specified. 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 **-bg**[*promotion_group*] [,*promotion_group*] ...

Brief report showing which revisions are associated with promotion groups. Optionally, specify which promotion groups to report on. Note, separate multiple promotion groups with commas.

-bl **-bl**[*user_id*] [,*user_id*] ...

Brief report of currently locked revisions. Optionally, specify which users to report on. Note, separate multiple user IDs with commas.

-bn **-bn**[*branch*]

Brief report showing the newest revisions on the trunk, or on a branch if specified.

-br **-br**{[*revision*] | [*from_rev***to_rev*] | [*branch*] | [*version*] | [*from_ver***to_ver*]}

Brief report of revision information. Optionally, limit to a specified revision, range of revisions, branch, version label, or range of version labels.

-bv **-bv**[*version*]

Brief report showing the revisions that correspond to a specified version label. If no version label is specified, it reports on the default version; if no default version is defined, it reports on all version labels.

-de **-de***end_date*

Limits report to revisions checked in within the specified date range ending at `end_date`. Use with the **-ds** option.

-ds **-ds***start_date*

Limits report to revisions checked in within the specified date range starting at `start_date`. Use with the **-de** option.

-f **-f***file*

Appends the report to the specified file rather than sending it to stdout.

-g **-g***group*

Limits the report to the specified promotion group.

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

-i Suppresses the indentation of branches when formatting the report.

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-l **-l**[*user_id*][,*user_id*] ...

Limits report to locked versioned items. Optionally, limit report to items locked by specified users. Note, separate multiple user IDs with commas.

-o **-o***user_id*[,*user_id*] ...

Limits report to versioned items owned by specified users. Note, separate multiple user IDs with commas.

-pp **-pp***project_path*

Sets the current project for this command execution. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Sets the current project database for this command execution. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If it is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

-r Reports on a specified revision or range of revisions, depending on usage:

- **-r**

  Limits report to the default revision. If no default revision is defined, it reports on the latest trunk revision.

- **-r***revision*

  Limits the report to the specified revision.

- **-r***from_rev***to_rev*

  Limits the report to the revisions between *from_rev* and *to_rev* (inclusive). If *from_rev* is omitted, the first revision at the bottom of the trunk or branch indicated by *to_rev* is assumed. If *to_rev* is omitted, the tip revision on the tip of the trunk or branch indicated by *from_rev* is assumed.

- **-r**{[*version*] | [*from_rev***to_rev*]}+

  Reports on all revisions and branches between the specified revisions. The range must start and end on the same branch or trunk.

-sp **-sp/@/***userID* | **Public**/*parent_workspace*/*child_workspace*

Specifies a public or private workspace. Note that user IDs are case-sensitive. To specify the Root Workspace, enter **/@/**RootWorkspace for the workspace value. If a workspace is not specified, the user's default workspace is used.

-v Reports on the revisions specified by a version label or range of version labels, depending on usage:

- **-v***version*

  Reports on the revisions specified by the version label.

- **-v***from_ver***to_ver*

  Reports on the revisions between the revisions identified by the version labels *from_ver* and *to_ver* (inclusive). If *from_ver* is omitted, the first revision at the bottom of the trunk or branch indicated by *to_ver* is assumed. If *to_ver* is omitted, the tip revision on the tip of the trunk or branch indicated by *from_ver* is assumed.

- **-v**{[*version*] | [*from_ver***to_ver*]}+
  Reports on all revisions and branches between the specified revisions. The range must start and end on the same branch or trunk.

-z Includes versioned files in subprojects.

Examples **Example 1:** The following example lists brief information (no archive details) on all revisions associated with the label REL_1.0 in and below the /chess project. Without the **Run -q** command, Vlog would print every filename processed, even those not matching the query.

```
pcli run -q Vlog -prC:\Users\All Users\Serena\VM\sampledb -idartd -br -
    vREL_1.0 -z /chess
```

Example 2: The following example lists all revisions checked in to versioned files in the /chess/client project by user kerstinb. Since the **-br** option is not specified, the archive details are printed for all files that have a matching revision.

```
pcli run -ns -q Vlog -prC:\Users\All Users\Serena\VM\sampledb -idAdmin -
    akerstinb /chess/client
```

Example 3: The following example lists all revisions that were checked in to versioned files in the /chess project between 4:30 P.M. and 5:00 P.M. on May 17, 1998. The **-br** option omits the archive details.

```
pcli run -ns -q Vlog -prC:\Users\All Users\Serena\VM\sampledb -idAdmin -
    br -ds"05/17/98 4:30PM"-de"05/17/98 5:00PM" -z /chess
```

Related Topics

For information about...	See...
Listing versioned files	"ListVersionedFiles command" on page 122

VmCopy command

Use the VmCopy command to copy projects or versioned items to a specified project.

Privileges required In the target project: Add or Remove Versioned Files (for just files), Copy Project (for entire projects). For more information on privileges, see the Administrator's Guide.

Alias VC

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax **VmCopy** [*destination options*] **-d***destination_entity* [*source options*] *source_entity ...*

Where:

-d*destination_entity* specifies the project to which you want to copy.

source_entity specifies the project or versioned items that you want to copy.

Options

-ac Copies the archives. The default is to use the existing archives.

-cc Copies the source configuration file.

-cn Creates a new configuration file.

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

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

A **-id** located before the **-d** switch applies to the destination project database. A **-id** located after the **-d** switch applies to the source project database.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

A **-id** located before the **-d** switch applies to the destination project database. A **-id** located after the **-d** switch applies to the source project database.

-pp **-pp***project_path*

Sets the current project for this command execution. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Sets the current project database for this command execution. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If it is not defined, then an error message is displayed and the command is aborted.

A **-pr** located before the **-d** switch applies to the destination project database. A **-pr** located after the **-d** switch applies to the source project database.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

-r **-r***revision*

Copies the specified revision rather than the entire archive.

-sc Copies the source access database.

-sn Creates a new access database.

-sp **-sp/@/***userID* | **Public**/*parent_workspace*/*child_workspace*

Specifies a public or private workspace. Note that user IDs are case-sensitive. To specify the Root Workspace, enter **/@/**RootWorkspace for the workspace value. If a workspace is not specified, the user's default workspace is used.

-t Copies only the tip revision rather than the entire archive.

-w Copies based upon the Version Manager 5.3/6.0 workfile hierarchy.

-z Includes versioned files in subprojects.

Examples **Example 1:** The following example creates a link in the destination project /ReadMe to the versioned files ReadMe.txt in the /bridge project and readme.html in the /chess/client project. This creates two shared archives that can be modified from either source location or from the new destination project. Note that the destination project must already exist.

```
pcli VmCopy -prC:\Users\All Users\Serena\VM\sampledb -idAdmin -d/ReadMe
    /bridge/ReadMe.txt /chess/client/readme.html
```

Example 2: The following example performs a baseline copy of all projects and files at and below the /chess project that have the version label REL_1.0, and creates new archives containing the revision matched by this label in the project /ChessRelease1 (this project must already exist).

```
pcli VmCopy -prC:\Users\All Users\Serena\VM\sampledb -idAdmin -d/
    ChessRelease1 -rREL_1.0 -z /chess
```

Example 3: As in Example 2, except the /chess project itself does not become a subproject of the /ChessRelease1 project. Since the /chess project contains a subproject named server, this example will copy that project to /ChessRelease1/server whereas the same project is copied to /ChessRelease1/chess/server in Example 2.

```
pcli VmCopy -prC:\Users\All Users\Serena\VM\sampledb -idAdmin -d/
    ChessRelease1 -rREL_1.0 -z /chess/*
```

Related Topics

For information about...	See...
Creating projects	"CreateProject command" on page 58
Creating project databases	"CreateProjectDB command" on page 61
Moving versioned files	"Move command" on page 128

While command

Use While to repeat commands in a loop

Use the While command to implement a typical scripting While loop. It allows scripts to repeatedly execute a set of commands as long as the result of a boolean command is true. A boolean command is a command that returns a status of true or false, where zero is true and false is any other status. Typically, the boolean command used is the Test command, which evaluates boolean expressions such as string comparisons.

Privileges required

None.

Alias

None.

Exit codes

Returns the value of the last command executed in the While loop when successful.

-2 PCLI command not found
-3 A non-PCLI related error or a command-specific error
-6 An invalid argument was specified
-7 An argument for a flag that is not needed
-8 A missing argument for a flag
-9 Wrong type was specified for an option's argument
-10 The specified file name cannot be read
-11 A required argument is missing
-12 A security exception occurred
-13 An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax

```
while [options] boolean_command
{
   action...
}
```

Where:

boolean_command specifies the name of the boolean command to use, typically the PCLI Test command. This command determines whether or not the following actions are executed.

action is the action to take in the loop. This action can be a PCLI command or any external program. Place each action on a separate line.

Options

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

Example

The following example goes through a list of directories and adds the files of all of the directories to one Version Manager project. The example first sets a variable array to the directories that contain the files that are to be added and then defines the While loop. The While loop first tests whether the number of the array element is less than the array size of the variable before preceding, lists the files in each directory into a file, and then adds the files to the release project using the AddFiles command. The external command, find, is a UNIX command.

```
set -a -vdirfiles /serena/uguide /serena/admin /serena/readme
set -vi 0
while test $i < $(arraysize dirfiles)
{
   run ->listfiles -e find $dirfiles[$i] -type f -print
   addfiles -pr/usrs/docs/vm -pp/release @listfiles
   calc -vi $i+1
}
```

Related Topics

For information about...	See...
Boolean command	"Test command" on page 160
Setting variables	"Set command" on page 142
Other commands used in scripts	ArraySize on page 52, Break on page 54, Calc on page 55, Continue on page 57, Echo on page 63, Exit on page 73, For on page 76, If on page 91, Readline on page 135, Return on page 138, and Test on page 160.

WhoAmI command

Use the WhoAmI command to report the active user ID used to access the specified entities.

Privileges required

None.

Alias

WAI

Exit codes

0	Successful command completion
-2	PCLI command not found
-3	A non-PCLI related error or a command-specific error
-6	An invalid argument was specified
-7	An argument for a flag that is not needed
-8	A missing argument for a flag
-9	Wrong type was specified for an option's argument
-10	The specified file name cannot be read
-11	A required argument is missing
-12	A security exception occurred
-13	An unknown problem

See "Appendix B: Exit Codes" on page 183 for more-detailed definitions of the exit codes.

Syntax

WhoAmI [*options*] [*entity*] ...

Where *entity* specifies a folder, project database, project, or versioned file.

Options

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

-id **-id***user id*[:*user_password*]

This option specifies a user ID and/or password for project databases and projects that have an access control database enabled. It overrides the value of the PCLI_ID variable for a single command. If VLOGIN, SSO/CAC (without CAC implemented), or LDAP is the login source, use **-id** to specify your user ID—and password too if a password is assigned to your user ID. For example, **-id***UserID*:*Password*
If VLOGIN, SSO/CAC, or LDAP is not the login source, your user ID is predetermined by your login source, but you must still use **-id** to enter your password, if you have one. For example, **-id:***PassWord*.

-idCAC_LOGIN **-id**CAC_LOGIN[:*PIN*:*Aliasname*]

If SSO/CAC is the login source and CAC is implemented, you can include the CAC PIN and Aliasname when you type the command.
For example,**-id**CAC_LOGIN:*PIN*:*Aliasname*. Or you can type just the CAC_LOGIN keyword and then answer the prompts for CAC PIN and Aliasname. For example, **-id**CAC_LOGIN.Or you can type the PIN and answer the prompt for the aliasname. For example, **-id**CAC_LOGIN:*PIN*.

-pp **-pp***project_path*

Sets the current project for this command execution. This option overrides the value of the PCLI_PP variable for a single command execution. If no project is specified, the PCLI_PP variable is used. For more information, see -pp on page 17.

-pr **-pr***project_database*

Sets the current project database for this command execution. This option overrides the value of the PCLI_PR variable for a single command execution. You must specify a project database for this command. If it is not defined, then an error message is displayed and the command is aborted.

-qe Quietly ignores nonexistent entities. If you specify an entity that does not exist, the command will act as if you had not specified the entity, and then proceed to any subsequent entities you may have entered.

-qz Ignore valid entities. If you do not specify an entity, or if you use the option **-qe** and all the specified entities are invalid, the command exits without throwing an error.

-sp **-sp/@/***userID* | **Public**/*parent_workspace*/*child_workspace*

Specifies a public or private workspace. Note that user IDs are case-sensitive. To specify the Root Workspace, enter **/@/**RootWorkspace for the workspace value. If a workspace is not specified, the user's default workspace is used.

Examples **Example 1:** The following example checks the current user ID on a project database that uses the HOST login source.

```
pcli WhoAmI -prC:\Users\All Users\Serena\VM\testpdb
```

Example 2: The following example passes a user ID to the WhoAmI command to see if that user ID will be used or if another login source has precedence in selecting the user ID.

```
pcli WhoAmI -prC:\Users\All Users\Serena\VM\sampledb -idAdmin
```

Related Topics

For information about...	See...
Adding users	"AddUser command" on page 50
Deleting users	"DeleteUser command" on page 70

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 180, 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 180.	254 (full path including the file name)
pvcs_bindir	As listed in "Prohibited Characters for Files and Directories" on page 180.	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 180, 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: Exit Codes

The following table lists and defines the PCLI exit codes.

Exit Code	Definition
0	Command completes successfully.
-1	At least one entity being used in the operation experienced a WARNING condition, such as a missing version label during a multi-file Get operation. The handler will save the earliest "worst" state, so an early WARNING (1) could be replaced by an UNKNOWN (-13), but not vice versa.
-2	The issued PCLI command or user-defined function cannot be found.
-3	1. A non-PCLI error, such as not being able to open a file for output because you do not have the correct UNIX permissions. 2. A command-specific error, such as tryihng to create a new project database in a directory that already contains a project database.
-6	An invalid option was specified for a command. For example, `echo -zz`.
-7	An argument was specified for a option that does not take an argument. For example, the `-z` option for the AddFiles command does not take an argument. So, if you specify an argument for the `-z` option (such as `-z/proj1`), this exit code is generated.
-8	An argument is required for a flag, but was not specified. For example, the `-pr` option requires an argument (the location of a project database). If you do not specify an argument, this exit code is generated.
-9	The wrong type is specified for an option's argument. For example, an index array for a variable must be an integer, and if a string index is specified, this exit code is generated.
-10	The file that was specified cannot be read.
-11	A required option for a command was not specified.
-12	A security exception occurred. You do not have privileges enabled for the command.
-13	An unknown problem. The PCLI cannot determine the error but knows an error exists.

Index

Symbols

Numerics

A

B

C

D

E

F

G

I

L

M

N

O

P

Q

R

S

T

U

V

W