



HelixCore

P4V User Guide

2017.3
October 2017

PERFORCE

www.perforce.com

© Perforce Software, Inc. All rights reserved.



Copyright © 1999-2018 Perforce Software.

All rights reserved.

Perforce Software and documentation is available from www.perforce.com. You can download and use Perforce programs, but you can not sell or redistribute them. You can download, print, copy, edit, and redistribute the documentation, but you can not sell it, or sell any documentation derived from it. You can not modify or attempt to reverse engineer the programs.

This product is subject to U.S. export control laws and regulations including, but not limited to, the U.S. Export Administration Regulations, the International Traffic in Arms Regulation requirements, and all applicable end-use, end-user and destination restrictions. Licensee shall not permit, directly or indirectly, use of any Perforce technology in or by any U.S. embargoed country or otherwise in violation of any U.S. export control laws and regulations.

Perforce programs and documents are available from our Web site as is. No warranty or support is provided. Warranties and support, along with higher capacity servers, are sold by Perforce Software.

Perforce Software assumes no responsibility or liability for any errors or inaccuracies that might appear in this book. By downloading and using our programs and documents you agree to these terms.

Perforce and Inter-File Branching are trademarks of Perforce Software.

All other brands or product names are trademarks or registered trademarks of their respective companies or organizations.

Any additional software included within Perforce Software is listed in "[License statements](#)" on page 178.

Contents

How to use this guide	9
Feedback	9
Other Helix Server documentation	9
Syntax conventions	9
1 Introduction	10
About P4V, the Helix Versioning Engine Visual Client	10
Basic concepts	10
Getting started with P4V	11
Checking for updates	11
2 Using P4V	12
Connecting to Helix Server	12
Favorite connections	14
Connecting to unicode-mode Perforce services	14
Creating and managing workspaces	15
Create a workspace	15
Change your workspace	19
View workspaces	20
Search for workspaces	20
Delete and unload workspaces	20
Defining a Workspace View	21
Navigating P4V	22
Terminology	22
Layout	22
Update status information	23
Modify views	23
About P4V icons	24
Bookmarking files	27
Bookmark a file or folder	27
Manage bookmarks	28
Using P4V shortcuts	28
Drag and drop	30
Searching and filtering	31
Find files in a depot or workspace	31

Find a file, folder, or specification in the active tab	32
Find specifications using filters	32
Save and reuse filters	33
Filter with file paths	34
Reconciling offline work	35
Exporting files	36
3 Configuring P4V	37
Configuring P4V preferences	37
Connections	38
Streams	39
Server Data	40
Behavior	40
Merge-Integrate	41
Integrate Flags	42
Copy	43
Branch	44
Double-click	46
Shortcuts	46
Logging	46
Display	47
Files and History	47
Features	48
Tools	48
Image Timelapse	49
File Editors	49
Diff	50
Merge	50
Applets	51
Editing user preferences	51
Viewing effective settings	53
4 Managing files	54
Adding files to the depot	54
Retrieving files from the depot	55
Editing files	55
Reverting files	56
Checking in files	57

View changelists	57
Submit changelists	58
Reverse a changelist submission	58
Restrict access to a changelist	59
Configure changelist display	59
Undoing changes	60
Displaying the revision history of a file or folder	61
Files	61
Folders	61
Changing a file's type	62
Renaming and moving files or folders	62
Cleaning up files and directories	63
Deleting files	63
Diffing files and folders	64
Diff dialog options	65
Check workspace consistency	65
View the state of the depot at a specified point in time	66
Diff large files	66
Shelving files	66
Shelve checked-out files in a pending changelist	67
Unshelve files	67
Submit shelved files	68
Delete shelved files	68
5 More file management tools	69
Viewing codeline history in the revision graph	69
Read the revision graph	69
Navigate the revision graph	70
Filter the revision graph	70
Display details	71
Viewing file history with Time-lapse View	71
Toolbar	72
Slider	73
Viewing image file history with Time-lapse View	73
Using the folder diff utility	75
Diffing text files	77
Navigate diffs	77

Set preferences	77
Diffing images	77
Diffing Microsoft Word files	78
Merging files	79
Navigate diffs	79
Merge text	79
Set preferences	81
6 Managing codelines	82
Creating branches	82
Merging files between codelines	83
Open files for merge	84
Resolving files	86
Resolve individual files	86
Resolve multiple files	87
Managing branch mapping	89
Work with branch mapping	89
Managing labels	90
Create labels	91
Label files	91
Display and search for labels	91
Edit labels	92
Delete and unload labels	92
Retrieve file revisions in a label	93
Display files associated with a label	93
Managing jobs	93
Create a job	94
Add a job to a pending changelist	94
View jobs	94
Search for jobs	94
7 Advanced P4V options	102
Configuring custom tools	102
Environment considerations	102
Add custom tools	102
Import and export tools	105
Launching P4V components from the command line	106
8 Working with streams	107

About streams	107
The mainline model	107
Stream views	107
Propagating change between streams	108
Stream depots	108
Terminology differences between command line client and P4V	110
About stream views	110
Stream path types	110
Stream path syntax	111
Inheritance between parents and children	111
Examples	112
Setting up streams	116
Creating new streams	117
Creating stream workspaces	118
Selecting streams	119
Using the stream graph	119
Display streams	119
Stream graph display conventions	120
Configure the stream graph display	121
Display stream status	122
Work in a stream	122
Branching with Streams	122
Merging down and copying up between streams	123
Merge down	124
Copy up	125
Propagate change between unrelated streams	125
Working with task streams	125
Overview	125
Create a task stream from a parent stream in the same depot	126
Create a task stream in a different depot	127
Create a task stream without a parent	128
Convert a task stream to a regular stream	130
Delete and unload task streams	130
Filter task stream files out of File History results	131
Working with virtual streams	131
View virtual streams	132

Stream path behavior in virtual streams	133
Create a virtual stream	134
Submit changes to a virtual stream	134
Merge and copy to a virtual stream	134
9 Using P4V for distributed versioning	135
Understanding DVCS and setting up the server	135
Init	135
Read this first	136
Directories and files	136
Add files	137
Clone	137
Prepare to fetch and push content between servers	138
Fetch and push	138
Configure security for fetching and pushing	139
Specify what to copy	139
What do Fetch and Push copy?	139
Resubmit	140
Branches	140
Understanding remote mappings	141
10 Integration with Swarm	143
Review workflow	143
Setting up the Swarm integration	144
Swarm integration features	144
Request a review	144
Update a review	149
Open a review in Swarm	155
Review ID and Review State columns	158
Glossary	160
License statements	178

How to use this guide

This guide tells you how to use P4V, the Helix Visual Client. It is intended for anyone using P4V to perform version control management tasks with Helix Server.

Feedback

How can we improve this manual? Email us at manual@perforce.com.

Other Helix Server documentation

See <https://www.perforce.com/support/self-service-resources/documentation>.

Syntax conventions

Helix documentation uses the following syntax conventions to describe command line syntax.

Notation	Meaning
literal	Must be used in the command exactly as shown.
<i>italics</i>	A parameter for which you must supply specific information. For example, for a <i>serverid</i> parameter, supply the ID of the server.
[-f]	The enclosed elements are optional. Omit the brackets when you compose the command.
...	<ul style="list-style-type: none">Repeats as much as needed:<ul style="list-style-type: none"><code>alias-name[[\$(arg1)... [\$(argn)]]=transformation</code>Recursive for all directory levels:<ul style="list-style-type: none"><code>clone perforce:1666 //depot/main/p4... ~/local-repos/main</code><code>p4 repos -e //gra.../rep...</code>
<i>element1</i> <i>element2</i>	Either <i>element1</i> or <i>element2</i> is required.

1 | Introduction

This chapter includes introductory topics to help you get started with P4V, the Helix Visual Client.

About P4V, the Helix Versioning Engine Visual Client	10
Basic concepts	10
Getting started with P4V	11
Checking for updates	11

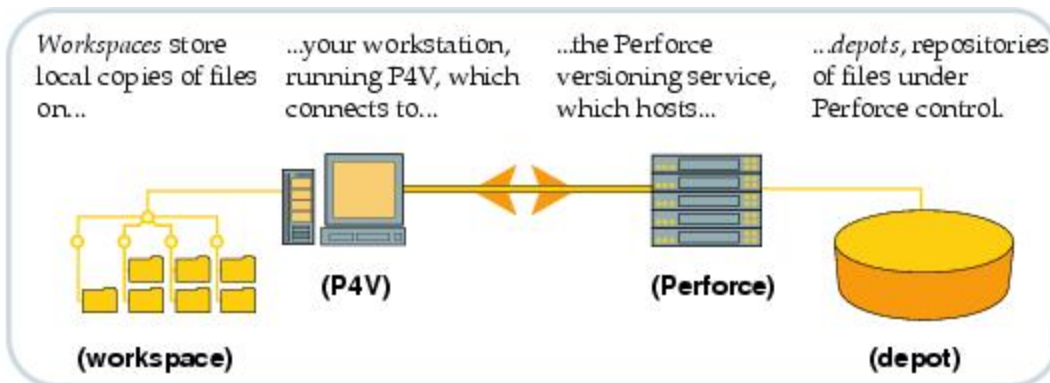
About P4V, the Helix Versioning Engine Visual Client

P4V, the Helix Versioning Engine Visual Client, is the cross-platform graphical user interface for Helix Server. You can use P4V on Windows, Mac, and Linux computers and benefit from an identical interface regardless of platform. To use Helix Server to manage files, you typically connect to the Helix Server versioning service using an application like P4V. P4V enables you to check files in and out, and perform various other versioning tasks.

Basic concepts

Perforce is an enterprise version management tool that you can use to manage source files and other documents, such as multiple revisions of a manual, web pages, or operating system administration files. The files managed by Perforce reside in a *depot*. To work on files, you open the files and edit them in your *workspace*. When you're done, you *submit* changed files to the depot using a *changelist*. The depot keeps track of all of the current and previous revisions of a file.

Perforce users connect to a shared file repository using a client application like P4V. P4V connects your computer to the Perforce versioning service and helps you move files between Perforce depots and your workspace:



The definitions for these Perforce terms are as follows:

- **Workspace:** folders or directories on your workstation where you work on revisions of files that are managed by Perforce.
- **Perforce application:** P4V (or another Perforce application, like the command-line client or P4VS, the Microsoft Visual Studio Perforce client), running on your workstation, which makes requests from the Perforce versioning service and delivers the results of those requests (files, status information, and so on) to you.
- **Service:** the Perforce versioning service responds to requests from Perforce applications, maintains depot files, and tracks the state of workspaces.
- **Depot:** a file repository hosted by the Perforce service. It contains all existing versions of all files ever submitted. Perforce can host multiple depots, but the examples in this guide show a single depot.

Getting started with P4V

To start using P4V, you must:

- Connect to a Perforce service (see ["Connecting to Helix Server" on page 12](#))
- Configure your client workspace (see ["Creating and managing workspaces" on page 15](#))
- Get files from the depot (see ["Retrieving files from the depot" on page 55](#))
- Add files to the depot (see ["Adding files to the depot" on page 54](#))

Checking for updates

To see if P4V updates are available, go to **Help > Check for Updates**. If an update is available, a dialog gives you the option to view the release notes and download it.

Note

This feature is available only when you connect to Helix Server version 13.1 or up. The Check for Updates feature may be disabled by your Helix Server administrator.

2 | Using P4V

This chapter describes the tasks you must perform to start working with P4V, along with tips for using the P4V user interface.

Connecting to Helix Server	12
Favorite connections	14
Connecting to unicode-mode Perforce services	14
Creating and managing workspaces	15
Create a workspace	15
Change your workspace	19
View workspaces	20
Search for workspaces	20
Delete and unload workspaces	20
Defining a Workspace View	21
Navigating P4V	22
Terminology	22
Layout	22
Update status information	23
Modify views	23
About P4V icons	24
Bookmarking files	27
Bookmark a file or folder	27
Manage bookmarks	28
Using P4V shortcuts	28
Drag and drop	30
Searching and filtering	31
Find files in a depot or workspace	31
Find a file, folder, or specification in the active tab	32
Find specifications using filters	32
Save and reuse filters	33
Filter with file paths	34
Reconciling offline work	35
Exporting files	36

Connecting to Helix Server

The first time you launch P4V, the Connection Setup Wizard (Mac: Connection Setup Assistant) runs automatically. You can use the wizard to specify connection settings and create a user and workspace if required. In subsequent P4V sessions, the **Open Connection** dialog is displayed by default. To run the wizard/assistant manually, go to **Connection > Set Up Connection...**

To connect to a Perforce service using a connection that you have used previously, do one of the following:

- If P4V is already running, go to **Connection > Open Recent** and select the connection.
- When you launch P4V, select the connection from the **Connections** drop-down in the Open Connection dialog. The Connections drop-down lists recent and [favorite connections](#).

Note

If your [Connection preference](#) is set to **Restore all previously opened connections** when you launch P4V, P4V opens the most recently used connection and skips the **Open Connection** dialog.

To connect to a Perforce service using a new connection:

1. Launch P4V or, if P4V is already running, go to **Connection > Open Connection...**
2. In the **Open Connection** dialog, enter the Perforce service name and port number for the connection using ***server_host:port_number*** syntax.

If your Perforce service is enabled for SSL (Secure Sockets Layer) encryption, use the following syntax: ***ssl:server_host:port_number***

Important

If you attempt to connect to an SSL-enabled Perforce service and you see a warning about an untrusted SSL connection or altered SSL fingerprint, contact your Helix Server administrator before completing the connection.

3. In the **User** field, enter your user name.
 - To browse for a particular user, click the **Browse...** button and select the user from that list.
 - To create a user, click **New...** and fill in the appropriate information.
4. (Optional) In the **Workspace** field, specify the name of your client workspace.
 - To browse for a particular client workspace, click the **Browse...** button and select the workspace from that list.
 - To create a client workspace, click **New...** and fill in the required information.
5. Click **OK**.

P4V connects to the specified Perforce service and displays a new instance of its main window.

Note

You can set Windows environment variables for Perforce connection settings, which makes the settings available to other Helix Server client applications (for example, P4EXP, the Helix Plugin for Windows Explorer).

- To set connection-related environment variables, go to **Connection > Environment Settings...** and specify the settings you want.

- To configure P4V to use environment connection settings at startup, go to **Edit > Preferences...** and, on the Connections page, enable **Open the connection that matches your Perforce environment settings**.

You can also click the **Change Settings** button on this page to set your connection-related environment settings.

Favorite connections

You can maintain a list of favorite connections and assign descriptive names to the entries. This frees you from having to remember port numbers and service host names.

- To add a favorite connection, go to **Connection > Favorite Connections > Add Favorite Connection...**
- To modify existing favorites, go to **Connection > Favorite Connections > Manage Favorites...**
- To connect to a favorite connection, go to **Connection > Favorite Connections**.

Connecting to unicode-mode Perforce services

The first time you connect to a Unicode-mode Perforce service, P4V requires you to choose a character set. If you are connecting to a Unicode-mode service, it is vital to configure this setting correctly to ensure that files are transferred properly between your client machine and the Perforce service. If you are unsure which setting to choose, consult your Helix Server administrator.

To configure a default encoding for Unicode-mode Perforce services, go to **Edit > Preferences...** (Windows) or **P4V > Preferences...**(Mac) and open the **Display** page.

If you are connecting to a Unicode-mode Perforce service for the first time, P4V displays the **Choose Character Encoding** dialog after you dismiss the **Open Connection** dialog. Specify the encoding you want and click OK. The encoding that you specify overrides any default that is configured, and the specified encoding is used when you subsequently connect to the same Perforce service.

Helix Server supports several variants of the UTF-16 character set, because the Windows, Mac, and Linux platforms differ in their handling of UTF-16 encoding, specifically, in the ordering of multibyte characters and the inclusion of a byte order marker (BOM) in the file header. The standard UTF-16 setting, **utf16**, is configured according to the typical defaults for your processor and is the recommended setting, unless you are certain that your client computer requires different byte-order/BOM settings.

For full details about configuring client and services to handle Unicode environments and files, refer to the Internationalization Notes (<http://www.perforce.com/perforce/doc.current/user/i18nnotes.txt>) for your version of Helix Server.

Creating and managing workspaces

A workspace specification defines the portion of the depot that can be accessed from that workspace and specifies where local copies of files in the depot are stored. This location is called the *workspace* (or *client*, in the P4 command line interface). A computer can contain multiple workspaces. A workspace is required when connecting to a Perforce service if you intend to work with files. You can connect using only host name and port, but you cannot exchange files unless you specify a workspace.

The mapping of depot files to local files is called the *workspace view*. If you are working with streams, the workspace view is generated by Perforce, based on the structure of the stream. If the structure of the stream changes, the workspace view is updated automatically. (In fact, you cannot manually edit the view of a stream workspace.) If you use classic depots, you must define and [maintain the workspace view](#) manually.

For more information about streams, see the "Streams" chapter in the [Helix Versioning Engine User Guide](#).



Create a workspace

To create a new workspace:

1. Open the **Workspace** dialog by doing one of the following:
 - Go to **Connection > New Workspace**.
 - Context-click a stream in the **Streams** tab and select **New Workspace**.
 - Context-click in the **Workspaces** tab and select **New Workspace**.
2. On the **Basic** tab, accept or change the default workspace name and root.

3. Map the workspace to the depot:

Classic depots: You can configure the workspace view (mappings) in the **Workspace Mappings** field by doing one of the following:

- Select the **View workspace mapping as text** icon  and enter your view specification using Helix Server client view syntax.
- Select the **View workspace mapping as tree** icon  and browse to the files and folders you want.

Build your workspace mapping by selecting a depot, folder, or file and using the Include, Exclude, and Clear icons; or context-click and select include, exclude, and clear options in the context menu.

For more information about mapping workspaces, see "[Defining a Workspace View](#)" on [page 21](#).

Stream depots:

- **Stream:** enter or browse for the stream that will be associated with this workspace.

If you enter a stream, the workspace view is populated automatically under **Workspace Mappings**; you cannot edit it.

Note

To dissociate a workspace from a stream, delete the entry in the Stream field.

- **Stream at change:** if you want to work using a stream definition as of a specific changelist, enter the changelist number here.

When a change is made to a stream definition, the stream is versioned using the current value of the change counter. Use stream at change when you want your stream workspace to use a view generated from the stream definition as of a prior changelist.

Using a stream-at-change view is useful if you need to work with a set of directories and files that are not identical to the set in the current stream. For example, your stream may no longer include certain libraries that had been included in an early version of the stream, but now you need those libraries to test a build. Enter the number of the last changelist to include those libraries. When you work in this workspace, P4V syncs to that changelist, allowing you to perform the test builds. Stream-at-change workspaces are read-only; you cannot check files into a previous changelist.

For more information about streams, see the "Streams" chapter in the [Helix Versioning Engine User Guide](#).

4. **Advanced options:** You can specify the following settings on the **Advanced** tab.

Owner	The user who created the specification.
Locked	If enabled, only the owner of the workspace can use, change, or delete the workspace specification.
Description	Your own explanation of the purpose of the workspace, or any related information you need to specify.
Host	(Optional) The computer where the workspace resides. To enable the workspace to be used from any machine, leave this field blank.
AltRoots	For workspace specifications used from hosts on different platforms, a list of workspace roots in host-platform-specific syntax.
File Options	<ul style="list-style-type: none">■ Allwrite: All files in the workspace are writable (can be modified).■ Clobber: Syncing files overwrites writable files on the workspace.■ Compress: Compresses data sent between the workspace and the Helix server.■ Modtime: Modification time for files edited in the client workspace is set to the time when the file is submitted to the depot.■ Rmdir: Deletes a workspace folder if all the files contained in the folder are removed.
Line ending characters for text files:	<p>The line-end convention used for storing text files on the workspace computer:</p> <ul style="list-style-type: none">■ Local: Uses the workspace platform default■ Unix: LF■ Mac: CR■ Win: CRLF■ Share: Line endings are LF. Any CR prior to a line ending is removed for storage or syncing (for disks shared between UNIX and Windows)

On submit:	Configures what happens when users submit files. The following options are available: <ul style="list-style-type: none">■ Submit all selected files: Default. All open files are submitted.■ Don't submit unchanged files: Files that have content, type, or resolved changes are submitted, Unchanged files are moved to the default changelist.■ Revert unchanged files: Files that have content, type, or resolved changes are submitted. Unchanged files are reverted.
Check out submitted files after submit	Select to cause submitted files to be reopened in the default changelist.
Client type	Specifies the type of client: writeable (the default), readonly , or partitioned . <p>Use readonly for short lived clients used in build automation scripts. Such clients cannot edit or submit files, but this should not be an issue in build scripts.</p> <p>Use partitioned to achieve the same thing as the readonly setting but with the additional ability to edit and submit files using that client.</p>

Note
Using writeable clients in build automation scripts can lead to **db.have** table fragmentation, which is used to track what files a client has synced. If you are experiencing such issues, use a read-only or partitioned client instead. A client of type **readonly** or **partitioned** is assigned its own personal **db.have** database table. The location of this table must first be specified with the **client.readonly.dir** configurable by an administrator.

Backup	<p>Not currently in use. Applies only to clients bound to cluster workspace servers.</p> <p>If a backup startup process is launched on a cluster workspace-server, by default all writable clients (see Type field) bound to that workspace-server are periodically backed up. To disable backup for a particular client, its owner must set the value of the Backup field to disable. Note: If the backup startup process is configured with auto-unload unused clients on, neither readonly clients nor those with backup disabled, will be considered for auto unload.</p> <p>For complete information, see version 15.1 of <i>Helix Versioning Engine Administrator Guide: Cluster Management</i>.</p>
---------------	---

5. Click **Save** to save your entries and create the workspace specification.

Change your workspace

Classic depots:

You can switch your workspace using the **Select Workspace** dialog. Open the Select Workspace dialog by doing any of the following:

- In the drop-down menu at the top of the Tree pane, click the drop-down arrow and select **Switch to Workspace**.
- Go to **Connections > Switch to Workspace**.
- Open the **Open Connection** dialog and click **Browse** next to the **Workspace** field.


You can also switch your workspace by context-clicking a workspace in the **Workspaces** tab and selecting **Switch to Workspace <workspace name>**.

Stream depots:

You can switch stream workspaces using the same methods as you use for classic workspaces. In addition, you can switch stream workspaces by doing the following:


- Context-click a stream and select **Work in this Stream**.
- Merge or copy to a stream: you are prompted to switch to the target stream's workspace.

In both cases, a warning dialog pops up and asks you to switch workspaces or create a new one. Click the **Switch Workspaces** button to switch your workspace. If more than one workspace is associated with the stream, the **Select Workspace** dialog opens, where you can search for and select the workspace you want.

However, if you have set your stream operations preference in the **Preferences** dialog to use the same workspace when you switch between streams, then P4V does not prompt you to switch workspaces. Instead the workspace view changes to include the stream you are switching to. In other words, the Stream field value in the workspace definition changes to the new stream. No matter what your stream operations preference, this is *always* the behavior in the Streams graph when you drag the Workspace  icon from your current stream to the one you want to work in.

View workspaces

To view workspaces for the server to which you are connected, do either of the following:

1. Go to **View > Workspaces** or select the Workspace  icon in the toolbar to open the **Workspaces** tab.
Click a workspace row to display the details of the client workspace specification.
2. Open the [Select Workspace](#) dialog.
Click a workspace row to display the details of the client workspace specification.

Search for workspaces

You can use filters to search for workspaces on the **Workspaces** tab and the **Select Workspace** dialog. You can filter workspaces by any combination of the following:

- **Owner:** select *current user* or enter a user ID.
- **Workspace Name**
- **Stream** name


You can also choose to **Show only workspaces available for use on this computer**.

For more information on filters, see "[Searching and filtering](#)" on page 31.

Delete and unload workspaces


Deleting a workspace removes the Helix Server record of the workspace but does not remove files from the workspace or depot. You cannot delete a workspace that has files checked out (open for edit) in P4V.

To delete a workspace:

1. Submit or revert any pending or shelved changelists associated with the workspace.
2. Go to **View > Workspaces** or select the Workspace  icon in the toolbar to open the **Workspaces** tab.
3. Context-click the label and select **Delete Workspace 'workspace_name'**.

Unloading transfers infrequently-used metadata from the versioning engine's database files to a set of flat files in an unload depot. If you unload a workspace, you can *reload* it if you change your mind and want to use it again.



To unload a workspace:

1. Submit or revert any pending or shelved changelists associated with the workspace.
2. Go to **View > Workspaces** or select the Workspace  icon in the toolbar to open the

Workspaces tab.

3. Context-click the workspace and select **Unload Workspace** '*workspace_name*'.

To reload an unloaded workspace:


1. Go to **View > Workspaces** or select the Workspace  icon in the toolbar to open the **Workspaces** tab.
2. Select the **Unloaded** icon  in the filter pane to open the Unloaded Workspaces dialog, where you can filter for and select unloaded workspaces to reload.
3. Context-click the workspace and select **Reload Workspace** '*workspace_name*'.

For more information about unloading, see the [P4 Command Reference](#).


Defining a Workspace View

The workspace view determines which portions of the depot are visible in your Workspace Tree and where local copies of depot files are stored in your workspace. If you use streams, the workspace view is generated and updated automatically. If you use classic depots, you must maintain the view manually, as described in this topic.

To define or change your workspace view:

1. Go to **View > Workspaces** or click the **Workspaces**  icon in the toolbar.
2. In the **Workspaces** tab, context-click the workspace and select **Edit Workspace**.
3. In the **Workspace** form, edit the **Workspace Mappings** field.
You can define the view *syntactically* and *graphically*, as described below.
4. When you have finished editing, save your changes.


Syntactic view specification

Click the **View workspace mapping as text**  icon and type your view specification using Helix Server client view syntax. Views consist of *mappings*, one per line. The left-hand side of the mapping specifies the depot files and the right-hand side specifies the location in the workspace where the depot files reside when they are retrieved from the depot. Example:

```
//depot/...      //bruno/depot/...
//user_depot/... //bruno/user_depot/...
//projects/...   //bruno/myprojects/...
```

For details about client view syntax, see the [Helix Versioning Engine User Guide](#).

Graphical view specification

Click the **View workspace mapping as tree** icon . The depot is displayed as a tree of folders and files. Context-click the file or folder you want to map and choose the mapping, as follows:

- **Include tree/Exclude tree:** include or exclude all files below the selected folder .
- **Include file/Exclude file:** include or exclude a specific file.
- **Include Special/Exclude Special:** use Helix Server syntax to specify the workspace view.

Alternately, double-click files or folders and use the resulting **Special Edit** dialog to define the view. This dialog enables you to specify options by clicking radio buttons or using the **Expressions** field to enter the left and right-hand components of a client view mapping.

Tip

To quickly add a depot path to the client view, go to **View > Filter Depot > Entire Depot Tree**, context-click the desired path and choose **Map to Workspace View**.

Navigating P4V

This section helps you get familiar with P4V terminology and layout and walks you through basic tasks such as updating status information, modifying views, and accessing administration tools.


Terminology

P4V menus and forms use a general approach to versioning terminology and actions, to ensure that that users with a variety of backgrounds can best understand what to do. If you have experience with P4Win or the P4 command line, note the following differences in terminology in P4V.

P4/P4Win term	P4V term
Client	Workspace
Sync	Get revision
Open for edit	Check out
Open for add/delete	Mark for add/delete

Layout


P4V displays one main window with two panes: the left pane, which is also called the **Tree** pane, and the right pane, which is where you do most of your work in P4V. These panes, as well as the other parts of P4V are described in detail below.

- **Toolbar and tooltips:** Across the top of the screen, there is a toolbar.
To get information about the items in the toolbar, as well as other Helix Server objects and P4V button, position the mouse cursor over the object without clicking. P4V displays a small window (tooltip) containing status or explanatory information about the object. P4V makes extensive use of tooltips.
- **Address bar:** this field, which is displayed under the toolbar, enables you to navigate to specific folders and files in the depot and in your workspace.
You can copy from and paste into the address bar.
- **Log pane:** The log pane, which displays the commands issued by P4V, is a tab located at the bottom of the P4V window.
To display this pane, choose **View > Log Pane**.
- **Dashboard pane:** The dashboard displays a variety of details about the status of your workspace and provides quick links for common tasks.
The dashboard is a tab located at the bottom of the P4V screen. To display the dashboard, choose **View>Dashboard**. To configure the tasks displayed in the dashboard, click the Settings button  that is displayed at the top right corner of the tab.
- **Context menu:** To display the context menu for an object on a **Mac**, option-click or click and hold. On **Linux and Windows**, right-click. (Note that this help system uses the platform-independent terminology "context-click" when instructing you to display a context menu.)

The left pane has two tabs: the **Workspace** tab shows the files on your computer, including files that are *not* in the depot. The **Depot** tab shows all of the files in the depot. To view the contents of a folder in the right-hand pane, click on that folder, and select **View > Files in Folder**.

The right pane contains tabs for working with changelists, labels, workspaces, users, jobs, streams, and branches. To display a tab, click the corresponding button on the toolbar or choose it from the **View** menu. At the bottom of the right pane, the Details tab displays details about the current selected object. To view multiple Details tabs (for example, to compare two objects), choose **View > Tear Off**.

Update status information

P4V indicates file information and status using a set of [icons and badges](#). To obtain updated status information and refresh the display, P4V periodically queries the Perforce service. To force a refresh operation, click .

Modify views

There are several ways to modify the way that information is presented in P4V's panes and tabs. At the top of these panes and tabs, you can see one or more of the following buttons:

 **Bookmarks**

Bookmarks act like shortcuts to locations in the tree. Set a bookmark by context-clicking on a directory and selecting **Bookmark**. You can then use this dropdown link to go to that location.

For more information, see ["Bookmarking files" on page 27](#).

Sort order

Use this drop-down button to sort the order of the list that you are viewing. The sort options are shown when you click the button.

Filter

This drop-down button allows you to filter your view in the following ways:

- Show or hide different types of files, such as deleted files and/or local files.
- Search for files and specs by various criteria and save those filters for reuse.
- In the Tree pane, "re-anchor" your tree to the bookmarks that you have created. If you select a bookmark from this list, that bookmark location becomes the top of your tree view.

For more information about filters, see ["Searching and filtering" on page 31](#).

Tear Off

This button allows you to "tear off" the current view. This is useful for comparing two different forms side-by-side. Once you have torn off a view, you can close the tear-off by simply closing that window.

File list view
















On the Files in Folder tab and the History tab (when viewing file history), use this icon to view files listed by file name or as thumbnails.















About P4V icons








P4V uses a variety of icons and badges (decorations) to depict Helix Server objects and their status. The following table describes commonly encountered icons.

P4V displays file icons in the Tree pane and throughout the user interface, with file status indicated as shown in the following table. Note that file icons might be displayed with multiple "badges" (for example, check marks, lock indicators), each indicating an aspect of the file's status. This table does not list all possible combinations. Note that red badges indicate actions taken by you, and blue badges indicate actions taken by another user. To display a tooltip containing more details about a file's status, hover the mouse pointer over the file.

Category	Icon	Description
Files		File in depot

Category	Icon	Description
		File in depot but not mapped by workspace view
		File in workspace differs from head revision
		File in workspace but not in depot
		File synced to head revision
		File synced to previous revision
		File needs to be resolved
		File locked by you
		File locked by other user
		File open for add by you (red "+")
		File open for add in other workspace (blue "+")
		File open for edit by you (red check mark)
		File open for edit by other user (blue check mark)
		File open for delete by you (red "x")
		File open for delete by other user (blue "x")
		File deleted in depot

Category	Icon	Description
		File open for rename/move ("x" indicates source, "+" indicates target)
		
		File open for branch
		File open for branch by other user
		File open for integrate (will need resolve)
		File imported from another stream (cannot be submitted to current stream)
		Symbolic link
Changelists		Shelved file in pending changelist
		Pending changelist has files that need resolve
		Pending changelist contains shelved files
		Pending changelist has no open files
Folders		(Blue folder) A folder in the Helix Server depot
		(Yellow folder) A folder in your client workspace
Depots		Spec depot: when enabled by your Helix Server administrator, a spec depot stores the history of changes to Helix Server specifications such as changelists.

Category	Icon	Description
		Remote depot: if configured by your Helix Server administrator, a remote depot maps a portion of another Helix Server repository as if it were a depot. Typically used to enable you to import and export third-party files without providing outsiders with access to your Perforce service.
		Stream depot: A depot where stream files are stored.
Workspaces		Workspace associated with a depot
		Workspace associated with a stream
Swarm		Pending changelist with shelved files in Swarm review
		Empty pending changelist with shelved files in Swarm review
		Submitted changelist with Swarm review

Bookmarking files

P4V enables you to create bookmarks so you can navigate quickly to often-used files and folders. You can organize the bookmarks using folders and separators. When you choose the bookmark from the list displayed under the **Tools > Bookmarks** menu item, P4V navigates to the corresponding file or folder and selects it, expanding any containing folders.

Bookmark a file or folder

1. In the depot or workspace pane, context-click the desired target file or folder. (P4V stores the location using local or depot syntax, depending on whether you select the target in the workspace or depot pane.)
2. Select **Bookmark...**
The **Bookmarks** dialog is displayed.

3. Specify the bookmark as follows:
 - **Name:** descriptive text to be displayed in the list of bookmarks
 - **Placement:** the location of the bookmark in the displayed hierarchy of bookmarks
 - **Location:** the path that specifies the location of the file or folder in the depot or workspace.
4. Click **OK** to dismiss the dialog and save your entries.

Manage bookmarks

1. Go to **Tools > Bookmarks > Manage Bookmarks...**
2. On the **Manage Bookmarks** dialog, you can create and edit bookmarks, create folders, and create separators.

Using P4V shortcuts

P4V supports the default shortcut keys listed below. You can also set up custom shortcuts. Go to **P4V > Preferences** (Mac) or **Edit > Preferences** (Windows) and set up your shortcuts on the [Shortcuts](#) page in the **Preferences** dialog.

Action	Mac	Windows
Display Workspace tab in Tree pane	+0	+0
Display pending changelists	+1	+1
Display submitted changelists	+2	+2
Display branches	+3	+3
Display labels	+4	+4
Display workspaces	+5	+5
Display jobs	+6	+5
Display Depot tab in Tree pane	+9	+9
Select all objects	+A	+A


Action	Mac	Windows
Copy text, path of selected file or folder, or contents of selected specification	+C	+C
Diff selected file	+D	+D
Check out file	+E	+E
Find	+F	+F
Close the current window	+F4	+F4
Go to a specification	+G	+G
Display Helix Server versioning service information in log window	+I	+I
Lock files	+L	+L
Create new specification	+N	+N
Open a new connection	+O	+O
Print selected item	+P	+P
Revert selected files	+R	+R
Submit selected changelist	+S	+S
Display Administration Tool	++A	++A
Copy depot path of selected file or folder to clipboard	++C	++C
Get latest revision	++G	++G

Action	Mac	Windows
Show in Stream Graph	++M	++M
Display Revision Graph of selected file	++R	++R
Display Time-lapse Display	++T	++T
Display revision history of selected file/folder	+T	+T
Unlock files	+U	+U
Paste text	+V	+V
Close the current window	+W	+W
Cut text	+X	+X
Redo last edit	+Y	+Y
Undo last edit	+Z	+Z

Drag and drop

P4V supports drag-and-drop shortcuts for common tasks. To enable or disable drag-and-drop file integration or diffing, go to **P4V > Preferences** (Mac) or **Edit > Preferences** (Windows) and set the drag-and-drop options on the **Behavior** page.

- **To add a file to Perforce**, drag it from a file browser (such as Windows Explorer) to a pending changelist.
- **To get a file revision** at a submitted changelist, workspace, label or file revision, drag the submitted changelist, workspace, label or file revision to the field next to the Get Revision dialog **To** field.
- **To integrate (merge) files**, drag the source folder to the target folder. The Merge/Integrate dialog opens, where you can refine your integration criteria.
- **To label a file**, drag a file or a folder to a label. Alternately, drag a label onto a file or folder.
- **To diff two file revisions**, drop a file or a file revision onto another file revision. To display file revisions, context-click the file and choose **File History**.
- **To diff revisions of different files**, display the File History for both files, then drag a revision from one window to the other and drop it on the other revision.

- **To diff two folder revisions**, drop a folder revision onto another folder revision. To display folder revisions, context-click the folder and choose **Folder History**.
- **To add files to a pending changelist**, drop files onto the pending changelist itself, or onto the Files field of the pending changelist's Submit form.
- **To move a file** from one changelist to another, drag files from the source Submit form to the target Submit form. Alternately, choose **Display>Pending Changelists**, and drag the files from the source changelist to the target changelist.
- **To filter the pending changelists, submitted changelists, labels, or jobs** by file path, drag files or folders from the Tree pane to the Filters pane in the specification tab.
- **To add a job to a changelist**, drag the job from the Jobs tab of the right pane to the Jobs field of the pending changelist. You cannot attach a job to a submitted changelist.
- **To locate a file in the depot**, drag the file from a pending or submitted changelist to the address bar (which is located below the P4V toolbar). Alternatively, copy the depot or workspace path (for example, select a file in a submitted changelist and choose **Edit>Copy**), then paste the path into the depot or workspace pane.
- **To switch panes**, drag any selected object to the toolbar button for the desired pane. For example, to add a job to a submitted changelist, drag the job from the Jobs tab to the submitted changelists button  P4V displays the Submitted Changelist tab. Drop the job on the target changelist.
- **For streams** drag-and-drop shortcuts, see ["Using the stream graph" on page 119](#).

Searching and filtering

You can search for files, changelists, workspaces, branch maps, streams, jobs, and labels using filters. Each of these has its own filtering procedure, but there are many shared search and filtering tools, which are described here.

Find files in a depot or workspace

To find a file in the depot or in your workspace:

1. Go to **Search > Find File...**
2. On the **Find File** tab, under **Search in**, enter the directory path you want to search. You can drag and drop the file path from the Depot or Workspace tree in the Tree pane.

3. Enter any of the following search criteria:
 - Under **Name matches any of the following**, select an operator (*contains, is, starts with, ends with*) and enter all or part of the file name. You can add multiple name match rows.
 - Under **Submission date or changelist**, enter a date, changelist, or range of dates or changelists.
 - Select **Include deleted depot files**.
4. Click **Find**.

Click a file in the search results pane to view file details.


Find a file, folder, or specification in the active tab

1. Click **Search > Find** or type +F (Windows) or +F (Mac).
2. Enter the search term in the **Find** dialog.

You can enter any part of a search term to retrieve results, unless you select **Match whole word only**. To search in a different tab, you must close the **Find** dialog, go to the new tab, and reopen the **Find** dialog for search.



Find specifications using filters

To enter filter criteria for most specification types:

1. Click the Filter disclosure triangle  **Filter:** in the upper pane of the specification (Submitted, Jobs, Labels, etc) tab.

2. Enter your filter criteria in the appropriate fields.

Use the following buttons to add or delete filter rows:

- To add conditions, click the plus  button.
- To remove conditions, click the minus  button.
- To nest conditions, enabling a switch in **Match** criteria (from *any* to *all*, for example), click the  button.

All retrieves results that meet all of the conditions you enter. It is equivalent to the logical operator "and." Use *All* to construct more restrictive searches. For example, if you want to retrieve only the jobs that contain both the term "installation" and the term "administration," use *All*.

Any retrieves results that meet any of the conditions you enter. It is equivalent to the logical operator "or." Use *Any* to construct less restrictive searches. For example, if you want to retrieve the jobs that contain at least one of the terms "installation" or "administration," use *Any*.

Note

Some fields may not be visible unless you expand the filter pane manually.


You can [save and reuse](#) filters, and you can also use the **Search** menu to initiate searches and select saved filters.

For more information about how to find and filter each specification type, see:

- For changelists, "[Checking in files](#)" on page 57
- For workspaces, "[Creating and managing workspaces](#)" on page 15
- For branch maps, "[Managing branch mapping](#)" on page 89
- For streams, "[Using the stream graph](#)" on page 119
- For jobs, "[Managing jobs](#)" on page 93
- For labels, "[Managing labels](#)" on page 90

Save and reuse filters

To save a filter:


1. Open the **Add Filter** dialog by doing one of the following:
 - In the filter pane, click the **Apply saved filters**  icon and select **Save Filter...**
 - Go to the Search menu and select **Save <spec type> Filter...**

2. On the **Add Filter** dialog, enter a name and folder location for your saved filter.


The default name is the filter expression.


3. Click **OK** to save.

To apply a saved filter:

1. In the filter pane, click the **Apply saved filters**  icon or go to the **Search** menu and select the filter type that you want.
2. Select a filter from the list.

To manage your saved filters:

1. Open the **Manage Filters** dialog by doing one of the following:
 - In the filter pane, click the **Apply saved filters**  icon and select **Manage Filters...**
 - Go to the Search menu and select **Manage <spec type> Filters...**
2. On the **Manage Filters** dialog, organize your saved filters by adding, deleting, or moving filters and folders.
3. Close the dialog to save your changes.

To clear a filter, click the **Clear filter**  icon or go to the **Search** menu and select **Clear <spec type>Filter....**


Filter with file paths

You can view jobs, changelists, or labels associated with particular files by entering the file path under **Files match any of the following file paths** in the tab's filter pane, or by using the **File Path Builder**. You can enter either a depot or workspace file path.

To enter file paths directly into the **Files match any of the following file paths** field, do any of the following:

- Use standard Helix Server file path syntax (**//depot/folder/folder/filename** or **//depot/folder/...**).
- You can use the standard Helix Server wildcards (* and ...) and revision specifiers (@ and #).
- For more information about wildcards and revision specifiers, see "[Issuing P4 Commands](#)" in the [Helix Versioning Engine User Guide](#).
- Drag and drop a file path from the Depot or Workspace tree into the field.
 - Click the drop-down arrow to view and select recent file paths.

To get help constructing a file path:

1. Click the **Construct a file path**  icon to open the **File Path Builder**.
2. Build a file path by selecting any one or combination of the following criteria:
 - **Folder** path in the depot or workspace.
Click **Browse** to view the depot and workspace trees and select a path.
 - **File** name or partial filename, using **contains**, **ends with**, or **starts with**.
 - **Revision range**:
 - All revisions.
 - Revisions starting or ending at a particular revision number, changelist number, label, workspace, or date/time.
 - Revisions ranging between two revision points (revision number, changelist, label, workspace, or date-time).
You can browse for changelists, labels, and workspaces.
As you enter or select values, the resulting file path appears in the **Path preview**: field.
3. Click **OK**.

The file path that you built appears in the **Files match any of the following file paths**:field.
All results that meet the search criteria appear in the search results window below.

Reconciling offline work

If for any reason you need to work offline, that is without having connectivity to Helix Server or without checking out files, you can manually enable write permission for the files and continue editing, adding, and deleting files as required.

To bring the Helix Server depot up to date with the work you did offline:

1. Context-click the folder that contains the files that you have edited, added, or deleted, and select **Reconcile Offline Work**.

2. If there are files that need to be reconciled, the **Reconcile Offline Work** dialog appears. P4V compares your workspace to the depot and lists the following in the dialog:
 - Files that were modified locally without being checked out. Select the files that you want to check out so that you can submit your changes.
 - Local files that are not in the depot. Select the files that you want to mark for add.
 - Depot files that are missing from your local workspace. Select the files that you want to mark for delete.

For renamed files, you must integrate the original file to the new filename and delete the original file. If you have altered the directory structure of your workspace, you might need to adjust your workspace view. For more information, see ["Renaming and moving files or folders" on page 62](#) and ["Defining a Workspace View" on page 21](#).
3. In the **Reconcile Offline Work** dialog, select the changelist that you want to add your changes to.
4. Click **Reconcile** to add the changes you selected in the dialog to the selected changelist.
5. Submit the changelist.

Exporting files

You can export files outside the current workspace mapping or if you have not set up or selected a workspace. For files mapped to a workspace, this option is not available.

Note

Exporting files may result in unexpected behavior. For example, if a file name contains spaces, P4V replaces them with underscores.

To export files:

1. Select **File > Export to**.
2. In the **Select folder to export to** dialog, specify the folder name and click **Select Folder**.

3 | Configuring P4V

This chapter describes how to configure your P4V and user preferences locally within P4V:

- "Configuring P4V preferences" below
- "Editing user preferences" on page 51

Users with *admin* privileges can also set:

- Performance-related P4V properties centrally using a P4JSAPI `centralsettings.js` file. For more information, see [Administering P4V Settings Centrally](#) in the *Javascript API for Visual Tools User Guide*.
- Performance- and feature-related P4V properties globally by running the `p4 property` command on the Helix server. This method is reserved for users with *admin* privileges. For more information, see [Configuring P4V settings in Helix Versioning Engine Administrator Guide: Fundamentals](#).

Note

Performance and feature-related properties set centrally or globally override local P4V settings. Some properties can only be set on the server level.

For information on using P4V with Swarm, see ["Integration with Swarm" on page 143](#). For more details on setting up the integration, done by an *admin* user on the Helix server, see [Swarm integration properties in the Helix Versioning Engine Administrator Guide: Fundamentals](#).

To view settings currently in effect, see ["Viewing effective settings" on page 53](#).

Configuring P4V preferences

To configure settings for P4V, do the following:

1. Go to **Edit > Preferences** (Windows) or **P4V > Preferences** (Mac).

The **Preferences** dialog includes the following configuration pages:

Connections	38
Streams	39
Server Data	40
Behavior	40
Merge-Integrate	41
Integrate Flags	42
Copy	43
Branch	44
Double-click	46

Shortcuts	46
Logging	46
Display	47
Files and History	47
Features	48
Tools	48
Image Timelapse	49
File Editors	49
Diff	50
Merge	50
Applets	51

2. Click **Apply** to save your changes or click **OK** to save your changes and exit the dialog.

Note

Local versus centralized preferences: Many P4V preferences can be defined or disabled centrally using the Perforce service. For example, an administrator can disable the **Labels** tab centrally, and you cannot override this setting in your local P4V preferences to make the **Labels** tab available. These centralized settings are specific to a Perforce service, so if you switch your connection to a different Perforce service during a P4V session, you may see different behaviors after you open the new connection. For example, if Perforce service A has enabled the **Labels** tab and Perforce service B has disabled it, the **Labels** tab will become unavailable when you switch your connection from service A to service B.

Note also that there are *performance-related* preferences (such as those on the **Server Data** page) that you can set centrally using the Perforce service. If these centrally-set performance preferences differ from your local preferences, your local settings continue to appear in the **Preferences** dialog even though the central preferences are overriding them.

Connections

You can configure the following settings for connecting to Perforce services:

When the application launches:

- **Show the Perforce Connection dialog:** Always prompt for connection settings when launching P4V.
- **Restore all previously opened connections:** Do not prompt for connection settings; reconnect to the Perforce service to which you were connected during your last session.

Opening and closing connections:

- **Use IP-specific tickets when logging in:** Specifies whether your login ticket is restricted to the IP address from which you are connecting.
- **Automatically log off when closing a connection:** Specifies whether your ticket is invalidated when you log out.

- **Don't expand Workspace and Depot trees to their previous state when opening connections:** Specifies whether the trees are initially displayed expanded only at the top (depot) level.

Streams

You can configure the way P4V handles operations such as branching and reparenting streams:

- **When branching streams, include file deletion actions (Server 12.1 or later):** Select to include files that were deleted from the parent stream when P4V populates the child stream. Equivalent to `p4 populate -f` command. The default behavior is to skip deleted files when branching to a new stream.
- **Don't allow streams to be reparented with drag and drop in the stream graph:** Disables drag and drop of streams inside the P4V stream graph. This is helpful if you find yourself accidentally reparenting streams while working within the stream graph.
- **Do not warn when checking out, adding, or deleting imported files:** Specifies the default behavior when operating on files that are imported into the current stream from another location. Choose **act on imported files** to have the files marked for edit, add, or delete as appropriate. Choose **ignore action for imported files** to have the file operations quietly ignored.

You can configure the way P4V handles workspaces when you perform stream operations with the following:

- **When clicking 'Work in this Stream':** Select **use different workspace** to have P4V prompt to create a new workspace for the selected stream if needs be. Select **reuse current workspace** to have P4V automatically switch your current workspace to the selected stream. Select this option if you regularly use streams with nearly identical workspace views and do not want to re-sync (retrieve) large amounts of redundant content each time you switch streams.
- **When dragging workspace icon to a new stream:** Select **use different workspace** to have P4V prompt to create a new workspace for the selected stream if needs be. Select **reuse current workspace** to have P4V automatically switch your current workspace to the selected stream. Select this option if you regularly use streams with nearly identical workspace views and do not want to re-sync (retrieve) large amounts of redundant content each time you switch streams.
- **Don't warn me when switching workspaces:** Specifies whether P4V switches workspaces silently when you switch streams.
- **Automatically update the workspace with all stream files when switching between streams:** Syncs (retrieves) files to your workspace when you switch between streams without being prompted.

You can control the display properties of the stream graph with the following:

- **Display stream:** Select the default name display and stream node width when viewing streams in the Stream Graph. To view the full depot path of the task stream in the Stream Graph, select **Stream Root**. Select **Name** to view the stream name only. You can override these defaults in Graph View Options on the Streams tab.

- **Stream width:** Adjust the slider to change the width of the stream nodes in the graph.
- **Show pending stream-to-stream merge and copy hints:** Enable to show which streams have changes to copy or merge. When disabled pending stream-to-parent merge and copy hints can be displayed by refreshing individual streams. Disabling this option reduces the amount of data P4V needs to request from the server.

Server Data

You can configure how much data P4V processes during a session to minimize server load for frequently-run commands and large data transfers. The following settings are available:

- **Check server for updates every *n* minutes:** Specifies how often P4V checks the Perforce service for updated file information. Frequent checks enable P4V to display current file status but increase the workload on the Perforce service.
- **Maximum number of files displayed per changelist:** Specifies the maximum number of files displayed in a changelist, to help minimize the time required to handle and submit very large changelists. This setting affects only the display of changelists and does not limit the number of files that a changelist can contain. Any files beyond the maximum number are displayed as a plain text list.
- **Maximum size of file to preview (excludes audio and video files):** Limits the size of image files displayed in the Preview tab on the Files pane, to limit the amount of image data sent from the Perforce service to P4V.
- **Number of changelists, jobs, branch mappings or labels to fetch at a time:** Specifies the number of specifications read in each batch fetched, to minimize server load and maximize P4V performance. To retrieve all entries, specify 0.
- **Disable parallel sync:** Disables concurrent transfer of files for all P4V connections. There is no configuration in P4V to turn *on* parallel sync; instead, parallel sync is enabled automatically when the `net.parallel.threads` config parameter is set in a server (2014.1 or later). Parallel sync can greatly reduce the amount of time it takes to update a workspace, especially when connecting across a high latency network. For more information on parallel processing, see [Using parallel processing for submits and syncs in *Helix Versioning Engine Administrator Guide: Fundamentals*](#).
- **Enable parallel submit:** Enables concurrent transfer of files when submitting to a 2015.1 or higher server. Similar to parallel sync, parallel submit can greatly reduce the amount of time required to submit by transferring multiple files across the network at the same time.

Behavior

You can configure the following general P4V user interface behaviors.

Prompts:

- **Warn before reverting files:** Specifies whether P4V displays a prompt before reverting files.
- **Warn when ignored files are not marked for add:** Select to view a warning when you attempt to mark a file in your workspace for add and it is included in an ignore list.

If you have created ignore lists and set the ignore list file name as the local **P4IGNORE** environment variable, P4V will not mark the files and file types listed in those ignore lists for add. The warning appears when you attempt to mark such a file for add or use the Reconcile process to add the file to the Helix Server depot.

For more information about ignore lists and **P4IGNORE**, see the [P4 User Guide](#).

- **Prompt for changelist when checking out, adding, or deleting files:** Specifies whether P4V displays the **Choose changelist** dialog when you open files.
- **Prompt to get latest revision when checking out files that are out of date:** Specifies whether P4V displays a prompt to get the latest revision when you attempt to check out files from an earlier revision.
- **Prompt when connecting to a new Perforce Sandbox:** Specifies whether P4V displays a prompt before you connect to a new Sandbox.

Drag and drop:

- **Enable integration on directory-to-directory drag and drop:** Launches the **Merge/Integrate** dialog when you drop a folder on another folder.
- **Enable diff comparisons on file-to-file drag and drop:** Launches the **Diff** dialog when you drop a file on another file.

Merge-Integrate

You can configure default behaviors for the Merge-Integrate dialog.

- Specify the default merge method that appears when the dialog opens:
 - **Specify source and target files:** The dialog prompts you to select source and target files
 - **Branch mapping:** The dialog prompts you to select a branch mapping
 - **Remember my last choice:** The dialog opens with the merge method you used the last time you opened the dialog

You can set different default merge methods depending on whether you open the Merge/Integrate dialog from a file or folder (non-stream) or a submitted changelist.

Note

You cannot set the merge method that appears by default when you open the Merge/Integrate dialog from a stream object or branch mapping, since the merge method for a stream object is always *Stream-to-stream* and the merge method for a branch mapping is always *Branch mapping*.

- Specify how to treat the files that you use to filter a branch mapping:
 - **Source:** The files that you include are treated as the source
 - **Target:** The files that you include are treated as the target
 - **Remember my last choice:** The dialog treats the files the way it did the last time you opened the dialog
- Specify which Options tab appears on top when the Merge/Integrate dialog opens:
 - **Resolve and Submit**
 - **Filter**
 - **Advanced**
 - **Remember my last choice**
- Specify the default Resolve and Submit options:
 - **Add files to pending changelist or Automatically submit after resolving**
 - **Automatically resolve files after merging** (select one of the resolve methods)
 - **Pending changelist:** Default or new
 - **Add previously linked job(s) to the new changelist**
- Specify whether to **Check for opened files and warn prior to merging**.

This option checks to see if any of the files selected for merging are open for other actions.

Click **Restore Defaults** to change the settings back to the P4V defaults.

For more information about these options, see [Merging Files Between Codelines](#).

Integrate Flags

Integrate flags can be applied when the Merge/Integrate and Branch dialogs run the **p4 integrate** command. You can configure these integrate flags to be applied by default.

- **Do not copy newly branched target files to workspace (-v):** Create a branch in the depot without retrieving the corresponding files from the depot to your workspace.
- **Enable baseless merges (-i):** Perform the integration even if source and target share no common ancestor, using the most-recently-added revision of the source file as the base.
- **Try to integrate changes when source deleted and re-added (-Di):** If the target file has been deleted and the source file has been changed, this option re-branches the source file on top of the target file. If the source file has been deleted and the target file has changed, this option deletes the target file. By default, outstanding edits cannot be integrated with a deleted file.
- **Force integration on all revisions, disregarding integration history (-f):** Integrate all revisions of source and target files, including revisions that have already been integrated. Typically used with a specified revision range.

- **Do not get latest revision of selected files (-h)**: Use the workspace revision of the target file. By default, the head revision of the target file is automatically retrieved into the workspace before integrating.
- **Disregard indirect integration history (-I)**: Restrict selection of the base file revision to direct (parent-to-child or child-to-parent) integration history.
- **Propagate source filetypes to target files (-t)**: Assign the target file the same file type as the source file (when integrating to existing target files).
- **Schedule 'branch resolves' instead of branching new target files (-Rb)**: Schedules a branch resolve, instead of branching new target files automatically.
- **Schedule 'delete resolves' instead of deleting target files (-Rd)**: Schedules a delete resolve, instead of deleting target files automatically.
- **Skip previously merged 'cherry-picked' revisions to improve merge results (-Rs)**: Skips cherry-picked revisions that have already been integrated. This option can improve merge results, but can also require multiple resolves per file.

These integrate flags are applied with different P4 commands, depending on the dialog, integration method, and Helix Server (P4D) version:

	Condition	Command	Available Flags
Merge/Integrate	Stream-to-stream method	p4 merge	None
Merge/Integrate	Branch mapping and Files and Folders methods	p4 integrate	All
Copy	Command available only with Helix Versioning Engine 2011.1 and up	p4 copy	-v
Branch	Helix Versioning Engine 2010.2 and earlier	p4 integrate	-v
Branch	Helix Versioning Engine 2011.1 and up	p4 populate	None

For more information about these P4 commands, see the [P4 Command Reference](#).

Copy

You can configure default behaviors for the Copy dialog.

- Specify the default copy method that appears when the dialog opens:
 - **Specify source and target files**: The dialog prompts you to select source and target files
 - **Branch mapping**: The dialog prompts you to select a branch mapping

- **Remember my last choice:** The dialog opens with the copy method you used the last time you opened the dialog

You can set different default copy methods depending on whether you open the Copy dialog from a non-stream file or folder or a submitted changelist.

Note

You cannot set the copy method that appears by default when you open the Copy dialog from a stream object or branch mapping, since the copy method for a stream object is always *Stream-to-stream* and the copy method for a branch mapping is always *Branch mapping*.

- Specify how to treat the files that you use to filter a branch mapping:
 - **Source:** The files that you include are treated as the source
 - **Target:** The files that you include are treated as the target
 - **Remember my last choice:** The dialog treats the files the way it did the last time you opened the dialog
- Specify which Options tab appears on top when the Copy dialog opens:
 - **Submit**
 - **Filter**
 - **Advanced**
 - **Remember my last choice**
- Specify the default Submit options:
 - **Add files to pending changelist** or **Automatically submit copied files**
 - **Pending changelist:** Default or new
 - **Add previously linked job(s) to the new changelist**
- Specify the default Advanced option:
 - **Do not copy newly branched target files to workspace (-v):** Create a branch in the depot without retrieving the corresponding files from the depot to your workspace.

Click **Restore Defaults** to change the settings back to P4V's defaults.

For more information about these options, see [Merging Files Between Codelines](#).

Branch

You can configure default behaviors for the Branch dialog.

- Specify the default branch method that appears when the dialog opens:
 - **Specify source and target files:** The dialog prompts you to select source and target files
 - **Branch mapping:** The dialog prompts you to select a branch mapping
 - **Remember my last choice:** The dialog opens with the branch method you used the last time you opened the dialog

You can set different default branch methods depending on whether you open the Branch dialog from a non-stream file or folder or a submitted changelist.

Note

You cannot set the branch method that appears by default when you open the Branch dialog from a stream object or branch mapping, since the branch method for a stream object is always *Specify source and target files* and the branch method for a branch mapping is always *Branch mapping*.

- Specify how to treat the files that you use to filter a branch mapping:
 - **Source:** The files that you include are treated as the source
 - **Target:** The files that you include are treated as the target
 - **Remember my last choice:** The dialog treats the files the way it did the last time you opened the dialog
- Specify which Options tab appears on top when the Branch dialog opens:
 - **Submit**
 - **Filter**
 - **Advanced**
 - **Remember my last choice**
- Specify the default Submit options:
 - **Add files to pending changelist** or **Automatically submit branched files**
 - **Pending changelist:** Default or new
- Specify the default Advanced option:
 - **Do not copy newly branched target files to workspace (-v):** Create a branch in the depot without retrieving the corresponding files from the depot to your workspace.

Click **Restore Defaults** to change the settings back to P4V's defaults.

For more information about these options, see [Creating Branches](#).

Double-click

You can select what you want to happen when you double-click various object types in P4V. Select a double-click behavior for an object by clicking in the **Double Click Behavior** column to open a drop-down list. The behaviors available for selection in the drop-down list depend on the object.

Click **Restore Defaults** to return all objects to their default setting, which is always the first selection in each drop-down list.

Shortcuts

You can enter keyboard shortcuts for commands used in the main P4V window, Revision Graph viewer, Folder Diff utility, and Time Lapse View.

To enter or edit a shortcut:

1. In the **Shortcuts for** drop-down, select the shortcut type:

- P4V
- Revision Graph
- Folder Diff
- Time-Lapse View

Only shortcuts for the selected window, viewer, or utility show in the shortcut list.

2. Click the **Shortcut** column for the command.

You may need to scroll the list window to the right to view the Shortcut column.

3. Enter the keyboard shortcut in the edit field.

Shortcuts must include either the command key, control key, or a function key

4. Click **OK**.

Some commands come with default keyboard shortcuts. You can restore all defaults by clicking **Restore Defaults**, or set an individual command to use a default by selecting the command row and clicking **Use Default**.

Logging

You can configure the following logging options:

Log pane options:

- **Show p4 reporting commands:** Specifies whether the log pane in the P4V window displays all commands issued by P4V, including commands issued by P4V to obtain status information from the Perforce service.
- **Show p4 command output for file operations:** For verbose log messages, enable this option.

Logging to a file:

- **Enable logging to file:** Logs P4V activity to the specified file.
 - **Name:** Specifies the name and location of the log file.
 - **Size:** Specifies the maximum size of the log file in kilobytes (KB).

Display

You can configure the following P4V display and localization options:

Application:

- **Show item count on tab bar of details pane:** For the Submitted tab, toggles display of the number of files and job fixes in the tab bar of the details pane (below the Submitted list).
- **Show the What's New in P4V page after upgrading:** Display the What's New page as a tab in the P4V window.
- **Dates:** Sets the date format used throughout P4V:
 - **OS format:** Use the format that your operating system uses.
 - **Perforce standard (yyyy/mm/dd hh:mm:ss)**
- **Show date and time as:** Specify whether to show the date and time for the shared versioning service (**Server time**) or for your local client.

Localization:

- **Language used for application menus, labels, and dialogs (requires restart):** Select English or, if it is available, Japanese.
- **Set encoding for all connections to:** Sets the character encoding for connections to unicode-mode Perforce services.

If you do not set the encoding here, you are prompted to enter the character encoding every time you set up a connection to a unicode-mode Perforce service. The encoding that you set here does not affect service connections whose character encoding has already been set at connection. If you are unsure which setting to choose, consult your Helix Server administrator.
- **Use the system character encoding in P4Merge when connected to non-unicode servers:** When enabled, overrides P4Merge's logic for determining character encoding (to preserve pre-2010.1 behavior).

Files and History

You can configure the way P4V displays files and file icons:

- **Use a distinct file icon for modified files:** Specifies whether P4V indicates files that you have edited after syncing them to your workspace.
- **Show Perforce filetype for files in the Workspace and Depot tree:** Toggles display of filetype in the tree panes.

- **Show revision information for files in the Workspace and Depot tree:** Toggles display of revision numbers in the tree panes.
- **Render thumbnails for Maya files:** Enables P4V to display thumbnail images of Maya media files.
- **File preview font:** Specifies the font that P4V uses to display files in the preview pane.
Click **Show fixed sized fonts only** to limit the selection in the font drop-down to fixed-sized fonts.
- **Hide files/revisions from 'task' streams (when following branch, copy actions):** Filters out file revisions that were submitted to task streams when you view file history

Features

You can enable or disable the following features. When you disable a feature, the tabs and dialogs associated with the feature no longer display in P4V.

Features:

- **Labels**
- **Jobs**
- **Streams**
- **Unload/Reload**
- **Distributed Version Control**

Menu options:

- **Merge, Copy and Branch Dialogs**
- **Set up Connection Assistant / Connection Wizard**
- **Sandbox Configuration**
- **Revision Graph**
- **Custom Tools**
- **Time-lapse**
- **Administration Tool**

All of the above are enabled by default. You must restart P4V for changes to take effect. For information about these features, see the appropriate section of this user guide.

Tools

You can configure the following Revision Graph and Time-Lapse View options:

Revision Graph:

- **Use software rendering for Revision Graph:** Enable only if your machine is having problems rendering Revision Graph output. For details, consult the [Perforce Knowledge Base](#).
- **Limit Revision Graph to ancestors and descendants:** Controls how much of a file's integration history is displayed.

Time-Lapse View:

- **By default Time-lapse view should show:** Specifies whether Time-lapse View displays only direct file history, branch (integration/merge) history, or information about the originating changelist for the selected file by default.

You can also select these display preferences in Time-lapse View. For more information about these options, see [Viewing Image File History with Time-lapse View](#). For more Image Time-lapse preferences, see "[Image Timelapse](#)" below.

Image Timelapse

Default revisions to retrieve for Image Time-lapse View: Specifies whether you want P4V to prompt you for the number of revisions to view when you use Time-lapse View, whether to limit the retrieved revisions to a certain number, or whether to view all revisions. You can improve performance by limiting the number of large image files retrieved at a time.

Image Time-lapse View Filetype Associations: Specifies whether you want to open files as images in Time-lapse view, and which image types to view as images. You can improve performance by limiting image rendering in Time-lapse view.

File Editors

To associate file types with the applications you use to edit them:

1. Click **Add**.
2. Select a file extension from the drop-down list.
3. Enter or browse for the associated application.
4. (Optional) Select **Always use the selected application to open files of this type** to set the application as the default.
5. Click **Save**.

You can enter as many applications as you like for each extension. All of the applications will appear as options when you context-click a file in P4V and select **Open With**.

Note

Any application that you've used to open a file from the context menu in P4V appears by default as an associated application on the File Editors page in the Preferences dialog, unless you remove it.

Diff

To set the default diff application, select one of the following:

1. **P4Merge**: The P4V companion diff tool.
2. **Other application**: Browse to your preferred diff tool.

To specify arguments for third-party diff applications, enter %1 for the name of the first file and %2 for the name of the second file in the **Arguments** field. P4V replaces these placeholders with the actual filenames when calling the diff application.

To assign diff applications by file type:

1. Click **Add**.
2. Select a file extension from the drop-down list.
3. Enter or browse for the associated application.
4. Specify arguments for third-party diff applications in the **Arguments** field:

Enter %1 for the name of the first file and %2 for the name of the second file. P4V replaces these placeholders with the actual filenames when calling the diff application.

5. Click **Save**.

The extension and associated application are displayed in the list of file type-application associations.

Merge

To set the default merge application, select one of the following:

1. **P4Merge**: P4V's companion merge tool.
2. **Other application**: Browse to your preferred merge tool.

To specify arguments for third-party merge applications, enter the following replaceable parameters in the Arguments field:

- Base file: **%b**
- Their/Source file: **%1**
- Your/Target file: **%2**
- Result file: **%r**

P4V replaces these placeholders with the actual filenames when calling the merge application.

To assign merge applications by file type:

1. Click **Add**.
2. Select a file extension from the drop-down list.

3. Enter or browse for the associated application.
4. Specify arguments for third-party merge applications in the **Arguments** field:

Enter the following replaceable parameters in the Arguments field:

- Base file: **%b**
- Their/Source file: **%1**
- Your/Target file: **%2**
- Result file: **%r**

P4V replaces these placeholders with the actual filenames when calling the merge application.

5. Click **Save**.

The extension and associated application are displayed in the list of file type-application associations.

Applets

Your administrator can enable Perforce services to serve applets that can run in P4V. In order for these applets to run in P4V, you must enable applets in your P4V preferences. For more information about Perforce applets, see the [Perforce Javascript API for Visual Tools User Guide](#) or consult your Helix Server administrator.

To enable Perforce applets to run in P4V:

1. Select **Allow Perforce applets to run in P4V**.
2. Specify the Perforce services from which you are willing to accept Perforce applets:
 - Enter the Perforce service name or host:port in the **Server** field.
 - Click **Add**.
3. Click **Advanced** to specify the following settings:
 - **Save cookies from applets** (Click **Remove Cookies** to delete all applet-generated cookies)
 - **Allow applets to store data locally**
 - **Manually configure web proxy used by applets for internet access**: Enter the web proxy address and port.

Editing user preferences

To configure your user preferences, go to **Connection > Edit Current User...**

Use the **Form** tab to configure the following:

- **Full Name:** Enter your full name as you want it to appear to other users.
- **Email:** enter the email address where you want to receive P4V notifications.
- **Job view:** enter the jobs that you want to appear automatically on all new changelists that you submit.

Use standard Helix Server filter expressions to specify the jobs. If you set the Job view field to any valid filter expression, jobs matching the Job view appear on any changelists you create. Jobs that are fixed by the changelist should be left in the changelist when you submit it; other jobs should be deleted from the changelist form before submission. For example, suppose the jobs at your site have a field called Owned-By. If you set the Job view field to **Owned-By=yourname status=open**, all open jobs owned by you appear on all changelists you create. For more information about how to use Helix Server filter expressions to retrieve jobs, see "[Managing jobs](#)" on page 93.

- **Reviews:** enter the files whose changes you want to review.

If your Helix Server administrator has activated the Helix Server change review daemon, the daemon will send you email notifications whenever files you have subscribed to in the Reviews field have changed. List files using depot syntax. For example, let's say you enter the following in the Reviews field:

```
//depot/main/...
//depot/.../README
```

The change review daemon sends you an email whenever any **README** file has been submitted and whenever any file under **//depot/main** has been submitted.

You can type depot paths directly in this field or use the **Reviews** tab to select depot paths interactively.

Use the **Reviews** tab to select depot paths for review:

- Expand the tree to select a folder.
- Context-click the folder name to view a menu of inclusion choices:
 - **Include or Exclude Tree:** include or exclude all files in all subfolders within the selected folder
 - **Include or Exclude Files:** include or exclude all files in the folder but none within subfolders
 - **Include or Exclude Special...:** open the **Special Edit of View Map** dialog, where you can specify filenames, extensions, or expressions written in standard Helix Server filter expression syntax.
 - **Clear:** clear your current review options for the selected depot folder. This option appears only if you have already used the Reviews tab to select depot paths for review.
- Alternatively, you can double-click the folder name to go directly to the **Special Edit of View Map** dialog, where you can specify filenames, extensions, or expressions written in standard Helix Server filter expression syntax.

To change your password, go to **Connection>Change Password...**

Viewing effective settings

If you want to view the settings currently in effect, do the following:

1. Select **Help > System Info**.
2. In the **System Info** dialog box, scroll down to the **Application** section, which lists all effective settings, along with any disabled features.

If nothing is listed, the user's local P4V preferences are used.

Following is a sample **Application** section:

```
Application:
P4V version: Helix Visual Client/NTX64/2017.2/1532340
Qt build library: 5.6.1
Qt runtime library: 5.6.1
Disabled features: Dashboard, UnloadReload, Jobs
Server refresh interval: 5 minute(s)
Maximum files displayed per changelists: 1000
Maximum file preview size: 100 k-byte(s)
Entries fetched at a time: 100
```

If the text **pushed from the server** occurs, this indicates that settings have been set using the **p4 property** command and are being read from the central properties table.

You can also set properties from the command line using the **p4 property** command. For example:

```
p4 property -l -n P4V.Features // List enabled/disabled features
p4 property -l -n P4V.Performance // List performance-related settings
```

4 | Managing files

This chapter describes how to manage files using P4V.

Adding files to the depot	54
Retrieving files from the depot	55
Editing files	55
Reverting files	56
Checking in files	57
View changelists	57
Submit changelists	58
Reverse a changelist submission	58
Restrict access to a changelist	59
Configure changelist display	59
Undoing changes	60
Displaying the revision history of a file or folder	61
Files	61
Folders	61
Changing a file's type	62
Renaming and moving files or folders	62
Cleaning up files and directories	63
Deleting files	63
Diffing files and folders	64
Diff dialog options	65
Check workspace consistency	65
View the state of the depot at a specified point in time	66
Diff large files	66
Shelving files	66
Shelve checked-out files in a pending changelist	67
Unshelve files	67
Submit shelved files	68
Delete shelved files	68

Adding files to the depot

To add a file to the depot, you must perform two actions: (1) open the file for add, which places the file in a changelist, and (2) submit the changelist, which copies the file to the depot.

To add a file to the depot:

1. In the **Tree** pane, click the **Workspace** tab.
2. Browse to the file you want to add.

If the file does not reside in the depot, its icon is unmarked ().

3. Context-click the file and choose **Mark for Add**.

The file icon displays a red plus sign () indicating that it is open for add.

4. To submit the changelist containing the open file, context-click the file and choose **Submit**.

The **Pending Changelist** form is displayed, listing the files in the changelist.

5. Enter a description of the change and click **Submit**.

The new file is added to the depot.

Note

If you do not see the local file in the Workspace tab, choose **Search > Filter Workspace > Show Local Workspace Files**.

Retrieving files from the depot

You can retrieve the most recent revision or any previous revision of a file from the depot to your workspace. In the Tree pane, open the folder containing the file you want to retrieve. The icons indicate the status of the files; see "[About P4V icons](#)" on page 24 for details.

Expanding a file revision that resulted from an integration shows the revisions for the file in the source codeline. Getting a revision from the source codeline does not affect the original file you are browsing.

To get the latest revision, context-click the file and choose **Get Latest Revision**.

To get a previous revision, context-click the file and choose **File History**. In the **History** tab, context-click the desired revision and choose **Get this Revision**. Alternately, context-click the file and choose **Get Revision...** In the **Get Revision** dialog, specify the desired revision.

To use a changelist number, label, workspace, or date to specify a revision, click **Specify revision using**, choose the desired method from the drop-down list, and specify the desired changelist number, label, workspace, or date.

To get that revision into your workspace even if the workspace already has the file, check **Force Operation**. This option does not affect open files.

Editing files

To edit a file:

1. In the Tree pane, find the file that you want to edit.

For details, see [Retrieving Files from the Depot](#).

2. Context-click the file and choose **Check Out**.

When you open the file for edit, it is placed in a changelist.

3. Using the editor associated with the file type, make your changes.

To launch the associated editor, double-click the file or context-click and choose **Open With**. (You might need to associate an editor with the file type.)

To define an editor: choose **Tools > Preferences**, click the **Editor** tab, and specify the desired editor for the file type.)

4. To place your revised version in the depot so other users can have access to it, context-click the file and choose **Submit**.

In the **Pending Changelist** dialog, enter a description of your changes and submit the changelist that contains the file.

To display a file without opening it for edit, double-click the file icon.

Reverting files

You can discard changes made to open files, reverting them to the revisions last synced from the depot. Reverting files also removes the reverted files from the pending changelist with which they are associated.

When you revert files that you:

- Marked for delete, they are reinstated in the client workspace.
- Marked for add, they remain in the client workspace.
- Merged or integrated, they are removed from the client workspace.
- Renamed or moved, only the file marked for add as part of the move operation can be reverted.

Note

Reverting a file that has been opened for edit will overwrite any changes you have made to the file since the file was opened.

To revert files:

1. Select one of the following:
 - One or more folders in the **Depot** or **Workspace** tree
 - One or more files in the **Depot** or **Workspace** tree
 - A pending changelist on the **Pending** tab, or one or more files within a changelist

2. From the **Actions** menu, select one of the following:
 - **Revert If Unchanged** (for files) or **Revert Unchanged Files** (for folders and changelists) to revert only files that have not changed (in terms of content or filetype) since they were opened.

The only files reverted are those whose workspace revisions are the following:

- Open for edit but have unchanged content and unchanged filetype.
- Open for integrate and have not yet been resolved.
- Open for add but are missing from the workspace.

Files open for add that are missing but that also have pending integrations will not be reverted.

- **Revert** (for files) or **Revert Files** (for folders and changelists) to revert all changes


Checking in files

When you mark files for add or delete, check them out, integrate them, or schedule them for resolve, the files are added to *changelists*. Helix Server changelists are lists of actions to be performed on files. The actions in the changelist are performed when you *submit* the changelist. *Pending* changelists are changelists that have yet to be submitted. Changelists are assigned unique numbers by the Perforce versioning service. In addition, a default changelist is maintained for each client workspace. If submission of the default changelist fails, the Perforce service assigns it a number.

View changelists

To display changelists, go to **View > Pending changelists** or **View > Submitted changelists**. P4V displays the **Pending** or **Submitted** tab in the right pane, which include a list view of changelists and details for selected changelists at the bottom.

To filter the displayed changelists, use the **Filter** expansion pane in either the Pending or Submitted tabs. You can filter by the following conditions:

- **User:** searches for changelists by the user who created them. Enter a user ID or *Current user*.
- **Workspace:** searches for changelists by the workspace used to create them. Enter a workspace name or *Current workspace*.
- **Files match any of the following file paths:** searches for changelists that include files in any of the paths that you enter. Drag the file from the Tree pane to populate the field with its file path, or click the **Construct a file path**  icon to open the **File Path Builder**.

For more information about file filters and the File Path Builder, see [Searching and Filtering](#).

Submit changelists

You can submit a changelist by context-clicking the file name in the Tree pane or by selecting a pending changelist from the **Pending** tab.

To check in individual files:

1. Context-click the files in the depot or workspace pane and choose **Submit...**
2. In the **Submit** dialog, enter a description, select files, and (optionally) attach jobs.
3. Click **Submit**.

To submit an existing changelist:

1. Go to the **Pending** tab.
2. Filter for and select the changelist by double-clicking it.
3. In the **Submit** dialog, enter a description, select files, and (optionally) attach jobs.
4. Click **Submit**.

To edit the description of a submitted changelist:

1. Context-click the changelist and choose **Edit Submitted Changelist**.
2. In the **Submit** dialog, edit the description.

Note

Only the submitter of a changelist can edit its description.

To move a file from one pending changelist to another:

1. Expand the source changelist.
2. Drag the file to the target changelist.

Reverse a changelist submission

You can restore the state of a file or folder in two ways:

- **Undo** restores a file or folder back to a specified changelist, date, revision number, or label. Any changes made after that point in time are not retained. For details, see ["Undoing changes" on page 60](#).
- **Back out** removes specific changes made at a given changelist, date, or revision but allows you to keep changes made in subsequent revisions.

To back out the effects of a single file revision or submitted changelist:

1. Right-click the revision or submitted changelist and select **Back Out**.

P4V prompts you to open the files in a new changelist.

2. Click **Yes**.

The files are opened in a pending changelist and restored to the state that preceded submission of the changelist that you are backing out. P4V restores file state as follows:

- **Edited files:** P4V restores the revision that precedes the one created when the changelist was submitted.
 - **Deleted files:** P4V restores the revision preceding deletion.
 - **Added files:** P4V deletes files that were added by the changelist.
3. To update the files with subsequent changes, resolve the files and merge in the desired changes.
 4. Submit the changelist.

Restrict access to a changelist

By default, all users can view a pending or submitted changelist, regardless of whether they are permitted access to the files in the changelist by the protections table. To restrict who can see a changelist, check the **Restrict Access to Changelist** option when editing a pending or submitted changelist.

Visibility of restricted changelists is as follows:

- **Pending changelists:** visible only to owner, regardless of whether other users have access to checked-out files.
- **Pending changelists containing shelved files:** visible to users who have access to one or more of the shelved files.
- **Submitted changelists:** visible to users who have access to one or more of the files that were submitted in the changelist.

Configure changelist display

To minimize the time it takes P4V to handle very large changelists, limit the number of files displayed in a changelist by doing the following:

1. Go to **P4V > Preferences** (Mac) or **Edit > Preferences** (Windows).
2. Click **Server Data**.
3. In the **Maximum number of files displayed per changelist** field, enter the number of files to display in a changelist.

You can still submit changelists with more than the specified number of files, but the file lists are displayed as follows:

- **Pending** and **Submitted** tabs display "There are # files in this changelist."
- **Details** tab displays the list of files in a simple text box (with no P4V file badges).

Undoing changes

You can restore the state of a file or folder in two ways:

- **Back out** removes specific changes made at a given changelist, date, or revision but allows you to keep changes made in subsequent revisions. For details, see ["Reverse a changelist submission" on page 58](#).
- **Undo** restores a file or folder back to a specified changelist, date, revision number, or label. Any changes made after that point in time are not retained.

The **Undo** option runs the `p4 undo` command. You select a revision to "undo to." This is also called the *selected* revision. As such, undo operates on a range from the head revision back to the selected revision +1. The operation undoes everything back to (but not including) the revision that is being "undone to," and the selected revision is preserved. This range-focused type of undo does not require any extra syncs or resolves: the **undo** command that P4V calls takes care of the entire operation.

The **Undo** option creates a range according to the limit type:

- For a revision number, the range goes from the selected revision +1 to the head revision.
- For a selected label under a directory selection, the label revision numbers for the various files are checked and undone starting at each file's revision +1.
- For a selected changelist, the range goes from the changelist number +1 to "now."
- For a selected date/time limit, the range goes from the date/time +1 (in seconds) to "now."

This works even if the +1 changelist or +1 date/time has no associated server entry. You can preview the undo operation before submitting your changes. You can also specify a revision as the keyword "have." In this case, P4V applies the range for the files under the directory selection according to each file's "have" revision +1.

For any limit type, if the selected file(s) are already at the head revision, there is nothing to undo.

To perform an undo operation:

1. Select one of the following:
 - A folder in the **Depot** or **Workspace** tree
 - A file in the **Depot** or **Workspace** tree
 - A file revision on the file's **History** tab
 - A label on the **Labels** tab
 - A submitted changelist on the **Submitted** tab
2. From the **Actions** menu, select **Undo**.
3. In the **Undo** dialog box, specify the revision, changelist, date/time, or label that you want to restore.
4. Specify whether to check out the changes to a new or existing changelist.
5. Optionally, to inspect the file revisions to be undone, click **Preview**.

6. Do one of the following:

- Click **Save to Changelist** to integrate the changes to be undone into the specified changelist without submitting them just yet.
- Click **Submit** to check out the files as specified and immediately submit them. This opens the **Submit Changelist** dialog box.

By default, the changelist description in the **Submit Changelist** dialog box displays a comment similar to the following: "Undo *<path to file or folder>* to revision 3."


Click **Submit** again to submit the changelist containing the files.

P4V submits the changes; they are now the new head revision in the depot.

Displaying the revision history of a file or folder

This section explains how to view the revision history of a file or folder.

Files

To display a file's revision history, context-click the file icon and choose **File History**. The **History** tab displays the file's history, including integration and label history. If a file revision is currently in your workspace, it is indicated with a red box:  .

To get a specific revision into your workspace, context-click it and choose **Get this Revision**.

To view a preview of a revision's content, select the **Preview** tab at the bottom of the Details pane. This shows a preview of any file at any revision. You can resize this tab to view a larger version of the file.

To display branching history, click  and choose the desired history from the drop-down menu.

If the file is an image, you can view thumbnails of the image by selecting **View > Show Files As** and selecting the thumbnail size that you want to view.

To compare two file revisions, click and drag one revision to the other. P4V launches the file diff utility and displays the differences.

Folders

To display a folder's revision history, context-click the folder icon and choose **Folder History**. The **History** tab displays the folder's revision history.

To compare two folder revisions, click and drag one revision to the other. P4V launches the [Folder Diff Utility](#).

Changing a file's type

Helix Server file types determine how a file is stored in the depot and synced to the workspace, and whether it can be diffed.

To change a file's Helix Server file type (or other storage attributes):

1. Context-click the desired file and choose **Check Out**.
The file is checked out.
2. Context-click the file and choose **Change Filetype....**
The **Change Filetype** dialog is displayed.
3. Set the desired type and attributes and click **OK** to dismiss the dialog.
4. Submit the changelist containing the file.

For details about file types and attributes, see "File Types" in the *P4 Command Reference*.

Renaming and moving files or folders

To rename or move a file or folder using P4V, you select the Rename/Move option on the object. You can also rename and move a file or folder in one operation.

To rename a file or folder:

1. Context-click the file or folder you want to rename and choose **Rename/Move**.
The **Rename/Move** dialog is displayed.
2. Specify the new name in the **New name** field.
You can either click **Submit** to submit the change immediately, or **Save to Changelist** to submit the renamed object at a later date.
The rename operation is not complete until you submit the changelist that contains it.

To move a file or folder from one location to another:

1. Context-click the file or folder you want to move and choose **Rename/Move**.
The **Rename/Move** dialog is displayed.
2. In the **New location** field, either type in the path of the new location or browse for it with the **Browse** button.
You can either click **Submit** to submit the move immediately, or **Save to Changelist** to submit the operation at a later date.
The object will not be moved until you submit the changelist that contains the move operation.

Cleaning up files and directories

To clean up the file system objects that are not under control of Helix Server or to restore workspace files to match the state of corresponding depot files, you can select the **Clean** option on a folder.

Warning

Changes performed by the **Clean** option are permanent. You cannot revert this operation.

To clean files and folders:

1. Select a folder and click **Actions > Clean**, or right-click a folder and select **Clean**.
If there are files that need to be cleaned up, the **Clean Workspace (Revert to Depot)** dialog appears.
P4V compares your workspace to the depot and lists the following files:
 - Files that were modified locally without being checked out
 - Local files that are not in the depot
 - Depot files that are missing from your local workspace
2. By default, all files are selected for cleanup. If there are any files that you do not want to clean up, clear the respective check boxes.
3. By default, files and directories listed in **P4IGNORE** files are excluded from cleanup and remain unaffected. If you do want to include such files, clear the **Apply P4IGNORE files for this workspace** check box.
In this case, P4V compares your workspace to the depot again and then also lists applicable files that were previously excluded in the respective section.
4. Click **Clean**.
5. In the **Confirm Deleting and Reverting Files** dialog, click **Continue** to confirm the operation.

Deleting files

To delete a file from the depot, you must mark it for delete, then submit the changelist containing the marked file. When you delete a file, a new revision marked "deleted" is stored in the depot and the file is removed from your workspace. Previous revisions in the depot are not affected.

To delete a file:

1. Context-click the file and choose **Mark for Delete**.
When you mark the file for delete, it is placed in a changelist.
2. Submit the changelist containing the file.

Diffing files and folders

To diff two files or file versions:

1. In the depot pane, select the two files you want to diff.
You can also select just one of the files, context-click it, and choose **Diff against...** to open the Diff dialog, where you can specify the file or file version you want to diff against.
2. Context-click and choose **Diff....**
The Diff dialog is displayed.
3. Specify the revisions of the files you want to diff and click **Diff**.
P4V launches [P4Merge](#), displaying the differences between the files at the specified revision.
Shortcut: drag the first file to the second file and drop it.

To diff two folders:

1. In the depot pane, select the two folders you want to diff.
2. Context-click and choose **Diff....**
The Diff dialog is displayed
3. Specify the revisions of the folders you want to diff and click **Diff**.
P4V launches the [Folder Diff Utility](#), displaying the differences between the folders at the specified revision.
The resulting display is based on your current client view. In the case of added and deleted files, the folder diff utility displays the location where the files would reside if they existed.
Alternative: you can context-click one folder, select **Diff Against...**, and select the folder you want to diff against in the **Diff** dialog.

To diff two folder revisions:

1. Context-click the folder and choose **Folder History**.
2. In the Folder History pane, click and drag one revision to the other.
P4V launches the [Folder Diff Utility](#).
Alternative: you can context-click the folder, select **Diff Against...**, and select the folder revisions you want to diff in the **Diff** dialog.

To diff two labels:

1. In the depot pane, select the topmost meaningful path.
2. Context-click and choose **Diff....** The Diff dialog is displayed.
3. For each of the two Path fields, click **Specify revision** and specify the labels you want to diff.
4. Click **Diff**. The folder diff utility displays the differences between the labels.
Shortcut: To launch a diff from the **Labels** pane, drag onelabel to another.

Diff dialog options

- **Path:** the two folders or files you want to diff.
If you choose **Workspace version on local disk**, you can ensure that all files in the workspace (including files within the client mapping that are not under Helix Server control) are displayed by using local syntax. To display only files under Helix Server control, use depot syntax.
- **Workspace version on local disk:** the file revision in your client workspace, including any changes you made after retrieving it from the depot and editing it.
- **Latest revision:** the revision that was most recently submitted to the depot (the head revision).
- **Have revision:** the revision you most recently retrieved.
Does not include any edits you made after retrieving it from the depot.
- **Specify revision:** enables you to designate the desired revision using a revision number, changelist number, date, label, or workspace.

Check workspace consistency

If you need to reconcile your workspace after working offline, see ["Reconciling offline work" on page 35](#).

To detect common inconsistencies between the files in your workspace and those in the depot, you can diff your workspace with the depot by selecting the folders of interest.

When diffing folders, the folder diff utility enables you to detect problems such as:

- Files that need to be added to or deleted from the depot (files you have created or deleted in the workspace)
- Files in the depot that need to be retrieved to your workspace (files that changed in the depot after you retrieved them)
- Files you have edited but not submitted.
- Local files that you changed without first opening them for edit or without setting write permission (for example, using `:w!<` in vi.)

The folder diff utility also detects client view mapping disparities such as:

- Unmapped or remapped files and directories
- Files and directories that were retrieved to the workspace and subsequently removed from the client view
- Files and directories that exist in the local workspace but are not in the current client view

To prevent or correct disparities that arise from changes to your client view, retrieve the affected files (choose **Get Latest Revision**).

View the state of the depot at a specified point in time

When you choose **Specify revision**, the corresponding pane displays the state of the depot at the specified date, changelist number, or label. This feature enables you to view the depot structure at that time without retrieving files, similar to P4Web's *Back in Time Browsing*.

Diff large files

To ensure correct results when comparing files that exceed 50,000 lines, you need to set the **diff.sthresh** configurable to be greater than the number of lines being diffed. By default, this configurable has a value of 50,000.

Note

This setup requires P4V and P4Merge 2015.1 or later.

To set the **diff.sthresh** configurable using the P4ENVIRO environment setting:

1. Set **P4ENVIRO**.

- On Windows platforms, set P4ENVIRO as a Windows environment variable (**Advanced system settings > Environment Variables**):

```
P4ENVIRO=%USERPROFILE%\p4envi ro .txt
```

- On Linux platforms:

```
export P4ENVIRO=~/.p4envi ro
```



2. Depending on your platform, create the file **%USERPROFILE%\p4envi ro .txt** or **~/.p4envi ro** containing the following line:

```
diff.sthresh=100000
```

This example is set to 100,000, but the actual value you choose should be greater than the average number of lines in the two files being compared.

For more information on the environment setting, see **P4ENVIRO** in *P4 Command Reference*.

Shelving files

Shelving enables you to store copies of open files temporarily in the shared Helix Server repository without checking them in. Shelving is useful for a variety of purposes, including taking and restoring snapshots of in-progress work and reviewing other users' code before it's checked in. When you shelve a file, a copy is placed in a pending changelist from which other users can unshelve it. Pending changelists that contain shelved files are displayed with the following icon and badge: . When the changelist is expanded, shelved files are listed under the **Shelved Files** node, indicated with the following icon: .

When you manage shelved files, note the following:

- **Basics:** To be shelved, a file must be checked out. However, you cannot *unshelve* a checked-out file.
- **Submitting shelved files:** As of Helix Server 2013.1, you can submit a shelved file directly. For previous versions of Helix Server, you must first unshelve a file to submit it, then delete the shelved copy. (Unshelving does not delete the shelved copy.)
- **Managing changelists:** You cannot move a shelved copy to another pending changelist. If you revert a file after shelving it, the copy remains shelved in the changelist until you delete it. Only the changelist owner can reshelve or delete files that are shelved in the changelist. For Helix Server releases that predate version 2013.1, you cannot submit a changelist that contains shelved files; you must delete the shelved copies before submitting. Starting with Helix Server 2013.1, you can submit shelved files directly, but your changelist must contain only shelved files.
- **File history:** No file history is created when you shelve or unshelve files.
- **Diffing:** You can diff shelved copies just as you diff any other file. For example, to display any changes you made after shelving a file, drag the shelved copy and drop it on the checked-out file.

Shelve checked-out files in a pending changelist

1. Context-click the changelist and choose **Shelve...**
P4V displays the Shelve dialog.
2. Check the files you want to shelve and click **Shelve**.
3. When prompted, enter a description and click **OK**.
P4V shelves the file in the selected changelist (or, if you are shelving files in the default changelist, creates a new changelist).

Alternately, you can shelve a checked-out file by dragging it from the pending changelist or depot pane to the **Shelved Files** node of the changelist where you want it shelved.

Unshelve files

After shelving a file, you or another user can *unshelve* it, which restores the shelved copy to your workspace and opens it in the changelist of your choice. Unshelving does not remove files from the shelf.

To unshelve files in a pending changelist:

1. Context-click the file changelist choose **Unshelve...**
P4V displays the **Unshelved** dialog.
2. Check the files you want to unshelve and click **Unshelve** and any other desired options.
The shelved file is copied to your workspace and opened in the specified changelist.

To unshelve a file that was shelved by another user, you must have permission to check out the file. When you unshelve a file that was shelved by another user, it is copied to one of your changelists, from which you can edit and submit the file.

To unshelve a file into a different branch than the one it was shelved in:

Note

Unshelving shelved changes into different branches or related streams is only available with Helix Server 2013.1 or later.

1. Select the **Map unshelved files** checkbox on the **Unshelve** dialog.
2. Select the way you want to map the unshelved files to the target branch:
 - **Using Branch Mapping:** Enter the branch mapping you want to use, or browse for it.
You can use any branch mapping that maps the branch the file was shelved in as either source or target. Your current workspace view must be mapped to the target.
 - **Using Stream:** Type or select the stream you want to use to unshelve.
The stream must be the child in relation to the location of the files being unshelved. For example, if the files are shelved in a mainline and you want to unshelve into a development child, you must select the development child. Likewise, if the files are shelved in the development child and you want to unshelve into the mainline, you must select the development child. This still holds true if you reuse your workspace across streams. If you are working in the mainline and shelve some files, then move your workspace to the development child, you must still specify the development child stream to unshelve.

Submit shelved files

As of Helix Server 2013.1, you can submit shelved files directly.

Note

If there are non-shelved files along with shelved files in a pending changelist, you must first revert the non-shelved files or move them to another changelist. You cannot submit shelved files from a task stream.

To submit shelved files in a pending changelist, context-click the changelist or shelved files folder and choose **Submit Shelved Files...****

Delete shelved files

Shelved files remain shelved until you delete them from the pending changelist. **To delete a shelved file from a pending changelist**, context-click the file and choose **Delete**. Alternately, context-click the pending changelist and choose **Delete Shelved Files**.

5 | More file management tools

This chapter describes P4V's advanced file management tools.

Viewing codeline history in the revision graph	69
Read the revision graph	69
Navigate the revision graph	70
Filter the revision graph	70
Display details	71
Viewing file history with Time-lapse View	71
Toolbar	72
Slider	73
Viewing image file history with Time-lapse View	73
Using the folder diff utility	75
Diffing text files	77
Navigate diffs	77
Set preferences	77
Diffing images	77
Diffing Microsoft Word files	78
Merging files	79
Navigate diffs	79
Merge text	79
Set preferences	81

Additional codeline management tools are described in "Managing codelines" on page 82.

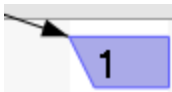
Viewing codeline history in the revision graph

The Revision Graph displays file integration history, showing when a file was added, branched, edited, merged, copied, or deleted.

To launch Revision Graph, context-click a file or folder in the Depot pane, Workspace pane, or Revision History dialog and choose **Revision Graph**.

Read the revision graph

Each revision of a file is represented by a shape. The shape denotes the action that created the revision. For example, the following shape indicates that the revision was created by branching the file:



When multiple revisions contribute to an integration, Revision Graph displays a bracket below the contributing revision, as shown in the following figure:



To display details about the meaning of the shapes and the lines that connect them, click the Legend tab in the lower right pane.

The top bar of the revision graph displays the changelist that created the file revision. **To view the changelist** (or sync to it or integrate it), context-click the changelist number.

Navigate the revision graph

To select revisions, click them or use the arrow keys. Details about the selected revision are displayed in the lower right-hand pane. **To select multiple revisions**, control-click (Windows) or command-click (Mac) them.

For files that have a large history, Revision Graph displays a portion of the graph in its main window, and a map of the graph in the lower left Navigator tab. A box in the Navigator outlines the portion displayed in the main window.

To navigate the diagram, do one of the following:

- Drag the box in the Navigator pane.
- Use the main window scrollbars.
- In the main window, use the mouse wheel or middle button.

To zoom in or out, move the slider in the toolbar or hold down the CTRL key and use the mouse wheel.

Highlighting shows the revisions that have contributed content to the selected revision or received content from it. **To highlight file revisions**, select the revision of interest and choose an option from the **Highlight** menu.

To diff two revisions, drag one revision to another or select the revisions, then context-click and choose **Diff Revisions**.

To move a line of revisions up or down: select it and click CTRL-up arrow or CTRL-down arrow.

Filter the revision graph

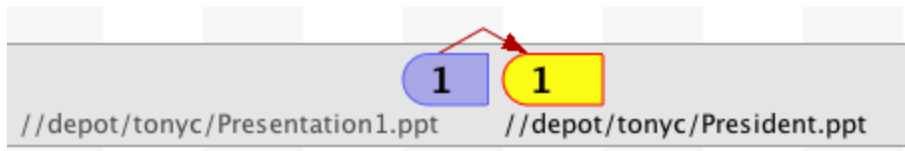
To reduce the detail displayed in the main window, you can filter the information. **To remove a file or folder from the main window**, uncheck it in the **File Filter** tree.

To enter a more precise file filter, click **Advanced...** and enter the file specification for the files and folders you want to retain in the main window (or, for files and folders you want to exclude, exclusionary lines preceded by "-"), check any filtering options you want to apply, then click **Filter**. To retain this filter in effect for future invocations of Revision Graph, click **Set as Default**.

To further compress the detail displayed in the main window, toggle the options on the **View** menu as follows:

- **File Renames Collapsed:** displays renamed files on a single line instead of multiple lines.
- **Compressed Integration History:** displays only revisions that were branched or integrated.

To compress file rename operations by omitting intervening revisions, choose **View > File Renames Collapsed**. Revision Graph displays the original and renamed file, indicating the operation with an angled arrow, as follows:



Display details

To **display details** about a file revision, click the revision in the main window. Details are displayed in the lower left pane.

Related revisions are listed on the **Integrations** tab. To **get the revision, diff it, or display its history**, context-click the revision on the **Integration** tab. To **view integrated revisions in the main window**, click the corresponding icon on the **Integrations** tab.

Viewing file history with Time-lapse View








Time-lapse View provides an interactive graphical representation of a file's history, showing when lines were added, changed, and deleted, who made the changes, and when the changes were made. Time-lapse View enables you to browse forward and back through changes dynamically, enabling you to locate changes of interest. Detail panes at the bottom of the window provide more information about selected chunks.




Note

Time-lapse view of image files works differently than described here. For more information, see ["Viewing image file history with Time-lapse View" on page 73](#).

To control the display, use the following settings and features.


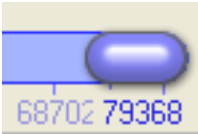

Toolbar

Mode	<p>Determines how many revisions are displayed. Options are:</p> <ul style="list-style-type: none"> ■ Single revision: one revision at a time is displayed ■ Incremental diffs: two adjacent revisions are displayed, with changes highlighted ■ Multiple revisions: a range of revisions is displayed, with changes highlighted
Content range	Specifies the starting and ending revision displayed.
Scale	Specifies the unit used: changelist number, date, or revision number.
User	Toggles display of the user that made the change.
	
Aging	Displays color coding to indicate how recently a change was entered. The darker the shading, the more recent the change.
	
Line numbers	Toggles display of line numbers.
	
Lifetimes	Toggles display of lifetimes, which are graphics that indicate by their width how long the adjacent chunk of text has been in the file.
	
Direct history	Display file revision history without reference to branching history.
	
Branch history	Display branching (merge/integration) history. Branch information appears above the timeline.
	
Originating changelist	Display information for the originating changelist for each revision. For a revision that has been integrated from another codeline, the branching history view provides changelist information only for the integration, not necessarily the changelist in which the content of the file was actually changed; that is, the <i>originating changelist</i> .
	

Find	Search text
	
Go to	In single revision mode, go to specified line number.
Next/previous diff	Go to next or previous change.
	
Line ending	Specifies how line endings and whitespace are treated to determine differences.
	

Slider

The slider enables you to browse rapidly through file revisions. The appearance of the slider corresponds to the mode you select. The unit by which the slider advances is specified by the mode you select (date, changelist, or revision). The revision, date, or changelist number is displayed under the slider.

Mode	Slider Appearance	Description
single revision		Move it to the right to display the next file revision or left to display the previous revision.
incremental diffs		Move it to the right to display the next pair of file revisions, or left to display the previous pair of file revisions.
multiple revisions		Move the right and left halves separately, to control how many revisions are displayed.

Viewing image file history with Time-lapse View

Time-lapse View for images provides an interactive graphical representation of an image file's history, showing changes, who made the changes, and when the changes were made. Time-lapse View enables you to view changes as a slideshow or browse forward and back through changes dynamically. Detail panes at the bottom of the window provide more information about selected revisions.

For information about Time-lapse View for non-image files, see "[Viewing file history with Time-lapse View](#)" on page 71.




You can view the following image file types in Time-lapse View:

- BMP
- GIF
- JPG, JPEG
- PNG
- PBM
- PGM
- PPM
- TIFF
- XBM
- XPM

To access Time-lapse View for images:

1. Context-click the image file and select **Time-Lapse View**.
2. In the **Time-lapse Image Range** dialog, select how many file revisions you want to view.
You can select a defined number of the most recent revisions; a range of revisions defined by revision number, changelist number, or date/time; or all revisions. You can also tell the system to remember your choice as the default to avoid being prompted again.
3. Click **OK**.

To control the Time-lapse View display, use the following settings and features.

Scale	Specifies the unit used: revision number, date, or changelist number.
Current selection	Specifies the current revision being displayed, in the unit selected in the Scale field.
Retrieved range	Specifies the starting and ending revision displayed, in the units selected.
	Enables or disables tweening effects, which generate intermediate frames between each image to give the appearance of a smooth transformation from one image to another.
Autoplay	Select to stop or start a slideshow presentation of the image revisions.
	
	

rotate images every seconds Specifies the time between image revisions during Autoplay.

2.00 

Slider Move the slider to the right to display the next file revision or left to display the previous revision. The revision, date, or changelist number is displayed under the slider.



Using the folder diff utility

To display the differences in the contents of two folders, you can diff them. For example, after [working offline](#), you can diff your workspace with the depot to determine how to submit your changes so that your workspace and the depot are consistent. To display the changes made to a folder over time, diff two versions of the same folder. The **Folder Diff** utility offers show/hide options that enable you to list only files of interest.

To diff folders:

1. Select two folders in the depot or workspace pane.
You can also select one folder and enter the second folder directly in the **Diff** dialog.
2. Right-click and select **Diff Against**.
3. In the **Diff** dialog, specify the paths and versions of the folders you want to compare.
4. Click **Diff** to launch the **Folder Diff** utility.

5. The **Folder Diff** utility lists the subfolders and files in both diffed folders, side-by-side.

Tip

You can use the following shortcut keys to expand or collapse a folder:

- Ctrl+Shift+Right Arrow (expand)
- Ctrl+Shift+Left Arrow (collapse)

Differences are highlighted as follows:



Violet highlight indicates content difference in file or subfolder.



Ivory highlight indicates file is present in only one of the two folders.

You can select from among the following display options (most of these can be used in combination):



	Show file pairs with identical content
	Show unique files, with no counterparts in the other folder
	Show file pairs with content differences
	Filter for the files that are displayed Click to select a saved filter or to create a filter using the Filter dialog. You can filter by filename (or part of a filename) and folder path. For more information about P4V filters, see " Searching and filtering " on page 31.
	Go to previous or next diff
	Show files in a tree hierarchy
	Show files as list

You can view details about a file by selecting it. The selected file is highlighted in a darker hue. File details appear in the following tabs below the diff windows:


- **Details**
- **Integrations**
- **Labels**
- **Preview**

You can also view a brief description of the latest change to any file by placing your mouse over the file icon.

Diffing text files




In its diff mode, P4Merge enables you to compare two text files to locate differences. (P4Merge can also [diff image files](#) and, if your Helix Server administrator has enabled it, [Microsoft Word .docx files](#).) The purple icon () is associated with the first file you selected, and purple bands highlight text that is in the first file but not the second file. The green icon () is associated with the second file you selected, and green bands highlight text that is in the second file but not the first file.

To display diffs in a single pane, go to **View > Single Pane Diff Layout**. In single-pane mode, deleted text is shown using strikethrough text. By default, P4Merge displays diffs in a side-by-side layout.


To toggle the display of line numbers, click .

Navigate diffs

To move forward and back through individual differences, or to locate specific text or lines, click the toolbar buttons listed below.

Go to next/previous diff	
Find text	
Go to line number	

Set preferences

To **configure P4Merge settings** to display or suppress differences in white space and line endings, click  or choose **File > Comparison Method**. To configure other settings, choose **Edit>Preferences** (Windows) or **P4Merge > Preferences** (Mac).

To **toggle the display of line numbers**, click .

Note

If you are comparing Unicode files, selection of the correct character set is essential. If you select the wrong character set, P4Merge cannot correctly compare files.

Diffing images

To **diff two different image files**: context-click an image file, choose **Diff Against...** and specify the other image, or (in the Depot, Workspace, or Files pane,) drop one image file's icon on another.

To diff two revisions of the same image file: context-click the image file and choose **File History**. In the File History pane, drop one of the desired revisions on the other revision.

P4V launches P4Merge, displaying the differences between the specified images. You can display image diffs side by side or overlaid.

To enlarge or reduce the displayed images, use the zoom sliders or choose the desired display option from the **View** menu.

To highlight differences, check **Highlight Differences** and use the adjacent slider to increase or decrease sensitivity. P4Merge highlights identical areas in gray and differences in yellow.

To blend or separate overlaid images, use the vertical slider: to blend the images equally, center the slider. **To increase the amount of an image used in the composite,** move the slider toward that image.

You can also view image revisions sequentially in Time-lapse View. For more information, see "[Viewing image file history with Time-lapse View](#)" on page 73

You can diff the following image file types:

- BMP
- GIF
- JPG, JPEG
- PNG
- PBM
- PGM
- PPM
- TIFF
- XBM
- XPM

Diffing Microsoft Word files

If your Helix Server administrator has enabled it, you can diff Microsoft Word **.docx** files in a P4Merge window using a rich comparison tool that shows differences in text, images, formats, styles, tables, headers, footers, and many other objects.




There are two .docx compare modes, **Parent-Child** and **Sibling**:

- **Parent-Child** mode is used to show how a file has changed from one version to another; this mode is set when you select **Diff Against Have Revision**.
- **Sibling** mode is used when there is no direct relationship between the files being compared; it is set, for example, when you drag and drop **.docx** files.

For more information about how to select files for diffing, see "[Diffing files and folders](#)" on page 64.

For more information about using P4Merge to view diffs, see the P4Merge help.



Merging files

If you and another user have edited the same file, Helix Server requires you to resolve those changes. One way of resolving is by merging your changes with the other changes using P4Merge. P4Merge enables you to compare two text files with a common base file to locate differences and to select the text that you want in the merged result file. The purple icon () is associated with the file that another user edited (their file), and purple bands highlight text that unique to that file. The green icon () is associated with file that you edited (your file), and green bands highlight text that is in the second file but not the first file. The base file is indicated by the yellow icon () and yellow highlighting indicates text that is in the base file but not in the other files.

In the top half of the window, P4Merge displays the base file surrounded by the two changed versions of the revision being merged. In the bottom half of the window, P4Merge displays the merge results file, where you select or enter the text that you want to check in. Make your changes as described below, and be sure to save them before exiting P4Merge.

Navigate diffs

To move forward and back through individual differences, or to locate specific text or lines, click the toolbar buttons listed below.

Go to next/previous diff	
Find text	
Go to line number	

Merge text

To merge files, you can either choose text from one of the files or enter changes directly into the merge results file.

Choose text chunks











To choose the entire contents of a file, click the corresponding icon in the merge diagram, as shown below.



To choose a **single chunk** of text from a set of diffs, click the corresponding button to the right of the chunk, as shown below.

```
# if defined(_SEQUENT_) || defined(__DGUX__) || defined(M_XENIX)
# define PORTAR 1
# endif
# if defined(_SEQUENT_) || defined(__DGUX__) || \
    defined(M_XENIX) || defined(__ISC)
# define PORTAR 1
# endif
# if defined(_SEQUENT_) || defined(__DGUX__) || defined(M_XENIX)
# define PORTAR 1
# endif
```

The buttons indicate the origin of the text and the way it was changed:

	Base file text
	Same in theirs and base.
	Same in yours and base.
	Text added to their file. Not in yours or base.
	Text added to your file. Not in theirs or base.
	Text deleted from their file.
	Text deleted from your file.
	Same in yours and theirs, different from base.
	Added to both yours and theirs.
	Deleted from both yours and theirs.

Note that the buttons are toggles: click once to select the corresponding text, click again to deselect it.

If P4Merge encounters too many changes to display all the required buttons, it displays this button:

 Click it to view the details.

Edit text

To edit the text in the merge result file, click on the text in the bottom pane and enter your changes. Manually-edited text is displayed with a gray highlight, like the middle chunk in the following example:


```

}

# else /* AIAMAG - RS6000 A


void
file_archscan( archive, func
char *archive;
void (*func) ();
{
    struct fl_hdr fl_hdr;

    struct {
        struct arx_hdr hdr;
        char pad[ 256 ];
    } ar_hdr ;

char buf[ MAXJPATH ];
1 --- -ff--

```

Set preferences

To configure **P4Merge settings** to display or suppress differences in white space and line endings, click  or choose **File > Comparison Method**. To configure other settings, choose **Edit>Preferences** (Windows) or **P4Merge > Preferences** (Mac).

To toggle the display of line numbers, click .

Note

If you are comparing Unicode files, selection of the correct character set is essential. If you select the wrong character set, P4Merge cannot correctly compare files.

6 | Managing codelines

This chapter discusses how to manage standard branching and merging workflow.

Creating branches	82
Merging files between codelines	83
Open files for merge	84
Resolving files	86
Resolve individual files	86
Resolve multiple files	87
Managing branch mapping	89
Work with branch mapping	89
Managing labels	90
Create labels	91
Label files	91
Display and search for labels	91
Edit labels	92
Delete and unload labels	92
Retrieve file revisions in a label	93
Display files associated with a label	93
Managing jobs	93
Create a job	94
Add a job to a pending changelist	94
View jobs	94
Search for jobs	94

For more information about viewing file history, including branching history, see "Displaying the revision history of a file or folder" on page 61 and "Viewing codeline history in the revision graph" on page 69.

For more information about using streams, which provide a very structured branching strategy, see "Working with streams" on page 107.

Creating branches

When you *branch* a codeline or stream, you create a copy of the source files and folders that is linked to the source by integration history. Typically you create a branch to enable concurrent work, perhaps to stabilize a release without impeding development or to permit experimentation without jeopardizing the stability of a mainline. Best practice is to keep the less stable branch up to date with its more stable neighbor by merging from it, then updating the neighbor with completed, stable work by copying. This approach ensures that no work is overwritten, and these policies are encoded into Helix Server [streams](#).

To branch a codeline or stream:

1. Select the source folders and files, context-click and choose **Branch**.

The **Branch** dialog is displayed.

2. In the **Choose target files/folder** field, specify the branch that you want to create. (Note: if the target already exists, you are not branching: you're **merging** — or *integrating* in the terminology of the P4 Command-line Client).
3. Configure any desired options (for details about options, see "[Merging files between codelines](#)" [below](#)) and click **Branch**. Files are opened for branch in a pending changelist.
4. If you've chosen the auto-submit option, the pending changelist is submitted immediately. If you've chosen **Add files to pending changelist**, submit the changelist when you are ready to create the branch.

Click **Set Defaults** to open the [Branch preferences page](#), where you can set default behaviors for the Branch dialog.

Note about Helix Server terminology: The term *branch* refers to a codeline, but is sometimes used as shorthand for a *branch specification* or *branch mapping*, which is a stored source/target mapping that is used to record and simplify branch operation.

Merging files between codelines

To create a new codeline (referred to as *branching*) or propagate a bug fix or new feature from one codeline to another, you *integrate* the required files to the target codeline. To integrate files, you open them for integration, specifying source and target, then submit the changelist containing the open files.

P4V performs three types of integration:

- *Branching*, which creates a new codeline or branch.
For more information, see [Creating Branches](#).
- *Merging*, which propagates change from one existing codeline to another and prompts you to resolve any conflicts between the codelines
- *Copying*, which is reserved for propagating a straightforward duplicate of the files in one codeline to another.

For more information about copying, see [Merging Down and Copying Up between Streams](#).

If you are merging changes into an existing codeline, you are required to *resolve* the files to ensure that you do not overwrite other users' changes and to preserve the file's revision history. Typical steps for propagating changes from one codeline to another are as follows:

1. Open files for merge.
2. Submit the changelist.
If there are conflicts, P4V notifies you and schedules the files for resolve.
3. [Resolve](#) the files, deciding how changed files are to be submitted.
4. Submit the changelist containing the resolved files.

When you open files for merge, you can specify the mapping of source to target using either a file specification or a branch mapping.

- **File specification:** you specify the source and target files when you open files for merging.
- **Branch mapping:** you select a branch mapping when you open files for merging. Branch mappings enable you to predefine source and target codelines. For details about branch mappings, see the [Helix Versioning Engine User Guide](#).

Note

The workflow for propagating change between streams ("merge up, copy down") is simple and specific. For more information, see [Merging Down and Copying Up between Streams](#).

Note

In the P4 Command-line Client, the term *integrate* is used not only to encompass all three integration types (branch, merge, copy), but is also used synonymously with the P4V term *merge*. Within P4V, *merge* can refer both to merging files from one codeline to another *and* to merging conflicts between files (the function performed by [P4Merge](#)).

Open files for merge

To open files for merging:

1. Select the source files and folders, then context-click and choose **Merge/Integrate**.
The **Merge/Integrate** dialog is displayed.
2. For Merge method, choose **Specify source and target files**.
The source files that you selected in step 1 are listed in the **Source files/folders** field.
3. Specify the target files and folders by typing or browsing to the destination.

4. To specify additional merge options, click the **Options** disclosure triangle.

The **Options** panel appears.

Specify integration options as follows:

Resolve and Submit Options: These options enable you to specify whether files are submitted manually or automatically, and to configure how conflicts are resolved.

Filter Options: Filtering enables you to choose a subset of the revisions that are queued for merging. For more information, see [Searching and Filtering](#).

Advanced Options: These options enable you to refine the results of the merge as follows:

- **Do not copy newly branched target files to workspace (-v):** Create a branch in the depot without retrieving the corresponding files from the depot to your workspace.
- **Enable baseless merges (-i):** Perform the integration even if source and target share no common ancestor, using the most-recently-added revision of the source file as the base.
- **Try to integrate changes when source deleted and re-added (-Di):** If the target file has been deleted and the source file changed, this option re-branches the source file on top of the target file. If the source file has been deleted and the target file has changed, this option deletes the target file. By default, outstanding edits cannot be integrated with a deleted file.
- **Force integration on all revisions, disregarding integration history (-f):** Integrate all revisions of source and target files, including revisions that have already been integrated. Typically used with a specified revision range.
- **Do not get latest revision of selected files (-h):** Use the workspace revision of the target file. By default, the head revision of the target file is automatically retrieved into the workspace before integrating.
- **Disregard indirect integration history (-1):** Restrict selection of the base file revision to direct (parent-to-child or child-to-parent) integration history.
- **Propagate source filetypes to target files (-t):** Assign the target file the same file type as the source file (when integrating to existing target files).
- **Schedule 'branch resolves' (-Rb):** Schedules a branch resolve, instead of branching new target files automatically.
- **Schedule 'delete resolves' (-Rd):** Schedules a delete resolve, instead of deleting target files automatically.
- **Skip 'cherry-picked' selected revisions (-Rs):** Skips cherry-picked revisions that have already been integrated. This option can improve merge results, but can also require multiple resolves per file.

- **Check for opened files and warn prior to merging:** Detect whether any of the selected files are open for other actions.

Note

P4V uses different P4 commands to apply these integrate flags, depending on the integration method:

- Stream-to-stream method: **p4 merge**
- Branch mapping and files and folders methods: **p4 integrate**

For more information about these P4 commands and flags, see the [P4 Command Reference](#).

5. To perform the merge, click **Merge**.


The specified files are opened for merge using any options you configured.

6. **Resolve** and submit the files.

Click **Set Defaults** to open the [Merge-Integrate preferences page](#), where you can set default behaviors for the **Merge/Integrate** dialog.

Resolving files

Conflicts occur when you attempt to merge a file into an existing codeline or submit a changelist containing a file that another user has edited and submitted while you had the file checked out. When the conflict occurs, Helix Server schedules the file for resolve. Conflicts must be resolved before you can submit the changelist that contains the conflicting file. To indicate a file that needs resolving, P4V

displays a question mark  badge.

When you attempt to submit a changelist containing a file that must be resolved, the Submit Changelist form displays instructions and the Submit button is grayed out. If the dialog says **Out of date** and there is a yellow triangle badge on any file, get the latest revision of that file by context-clicking it and selecting **Get Latest Revision**. This will not overwrite the copy of the file that is in your workspace. After you have the latest revision, you can resolve the file. You can resolve files [individually](#) or attempt to resolve [multiple files at once](#).

You can resolve conflicts resulting from content changes, filetype changes, attribute changes, moves, deletes, and branching.

Resolve individual files

To resolve an individual file:

1. Context-click the file and select **Resolve...**
2. The **Resolve** dialog opens in interactive mode by default.

If the file requires multiple resolve types (if, for example, another user has checked in changes to both the content and filetype since you checked the file out), you can select **Auto resolve multiple files**, but the default is to **Interactively resolve files one at a time**. In this case, each resolve type is listed as a separate row in the **Files to resolve** list.

For each file/resolve-type combination, P4V recommends an action, based on the differences and conflicts in the file selected. This recommendation is followed by an explanation. The action button for the option is highlighted in blue.

3. (Optional) For more information about the files being resolved, select an option from the list of **Additional Actions**:
 - **Open File**: Enables you to open either version of the file individually or the merged result file in any editor.
 - **Diff**: Enables you to diff the files with each other, with the base file, or with the merged result.
 - **File History**: Displays the revision history of either file.
 - **Time-lapse View**: Displays the history of either file using the [Time-lapse View](#) tool.
 - **Revision Graph**: Displays the codeline history of either file using the [Revision Graph](#) tool.
4. Select whether to **Merge binary files as text when resolving content**.

If you select this option, P4V treats binary files like text files and attempts a textual merge between the source and target files.
5. Follow the recommended action (highlighted in blue), or select another resolve option:
 - **Accept Source**: Replaces the copy of the file in your workspace with the version that is in the depot, discarding your changes.
 - **Accept Target**: Accepts the file that is in your workspace, overwriting the version that is in the depot when you submit the file.
 - **Accept Merged**: Replaces the file in your workspace with the merged result of the two files listed in the box at the top of the screen.
 - **Run Merge Tool**: Open the merge tool to edit the file and save the merged result (P4Merge is the default merge tool, but you may have chosen another tool in [P4V Preferences](#)).

As each file is resolved, it is removed from the list of **Files to resolve**.
6. To check in the changes, submit the changelist that includes the resolved files.

Resolve multiple files

To resolve multiple files at once, automatically:

1. Select the files, context-click, and select **Resolve...**
2. Select **Auto resolve multiple files**.

The **Resolve** dialog displays the **Files to Resolve**. As files are resolved, they are removed from this list.

To limit the files being auto-resolved to a particular resolve type or file type, select the type from the **Select** drop-down. This can be helpful when different resolve types might require different auto-resolve actions, or when some resolve or file types require you to resolve them interactively. For example, if your list of files to resolve includes both binary and text files, you might select the text files for auto resolve and leave the binaries to be resolved interactively.

The dialog displays the number of selected rows immediately above the list.

3. Select whether to **Merge binary files as text when resolving content**.

If you select this option, P4V treats binary files like text files and attempts a textual merge between the source and target files.

4. Select a **Resolve method**:

- **Safe automatic resolve (no merging)**: Accepts the source file (the file in the depot) if it has the only changes. Accepts the target file (the file in your workspace) if it has the only changes. Doesn't resolve if both the source and target have changed.
- **Automatic resolve (allow merging)**: Accepts the source if it has the only changes. Accepts the target file if it has the only changes. Merges changes if both the source and target have changed and there are no conflicts.
- **Accept Source**: Replaces the copy of the file in your workspace with the version that is in the depot, discarding your changes.
- **Accept Target**: Accepts the file that is in your workspace, overwriting the version that is in the depot when you submit the file.
- **Automatic resolve (allow merging with conflicts)**: Accepts the source if it has the only changes. Accepts the target file if it has the only changes. Creates a merged file if both the source and target have changed, even if there are conflicts. Where there are conflicts, both versions are included with text notations indicating the conflicts.

5. (Optional) Select **Set as Auto Default** to set your selections as the default for auto-resolving multiple files.

6. Click **Auto Resolve**.

If you selected an auto resolve option that does not allow merges when there are conflicts, any conflicts will prompt P4V to recommend that you resolve the files interactively one at a time.

7. (Optional) Repeat if you are auto-resolving multiple resolve types separately.
8. To check in the changes, submit the changelist that includes the resolved files.

To resolve multiple files one at a time (recommended when there are conflicts):

1. Select **Interactively resolve files one at a time**.
2. Follow the procedure described in [Resolve individual files](#).

Managing branch mapping

In Helix Versioning Engine, a *codeline* is a set of related files — for example, all the source code required to build your software product. Copying an edit from one file set to the other is called *merging* or *copying*. Copying a set of files to create a new codeline (or equivalent) is called *making a branch* or branching. Branching is performed using the **Merge/Integrate** feature. You can perform simple branches using a *file mapping*.

To ensure that complex branching is done in a controlled manner (for example, to prevent typographical errors when entering target directories or to make sure that complicated branches are performed correctly), you can create a *branch mapping*, which specifies the relationship between two codelines. When you branch, you can use the branch mapping instead of a file mapping. Branch mappings are displayed in the right pane on the **Branch Mapping** tab.

Note

Helix Server Streams provide an alternative approach to managing codelines. For more information, see the "Streams" chapter in the *Helix Versioning Engine User Guide*.

Work with branch mapping

To create a branch mapping, choose **File > New > Branch Mapping** and enter the required information. To prevent the mapping from being changed by other users, check **locked**. To confine integration to closely related files, choose **direct**; to enable integration between distantly related files, choose **indirect**.

Use the **View** field to reflect the relationship between source and target codelines. For example, to create release 2 of Jam from the code in your main codeline, you might use the following view:

```
//depot/jam_proj/... //depot/jam_r2.0/...
```

To view a branch mapping, go to **View > Branch Mappings** or click the **View branch mappings**  button in the toolbar. Use the **Filter** pane at the top of the **Branch Mappings** tab to search for mappings by owner and branch mapping name (or part of the name). Double-click a branch mapping to view it.

To change a branch mapping, double-click it in the **Branch Mappings** tab to open the **Branch Mapping** dialog. Click **Edit** to enter your changes.

To delete a branch mapping, click the mapping you want to delete, then select **Edit > Delete Branch branchname**.

To integrate using a branch mapping:

1. Right click the files you want to integrate and choose **Merge/Integrate**.
The **Merge/Integrate** dialog is displayed.
2. Select *User branch mapping* as your **Merge method**.
3. Browse for the branch mapping you want to use. Select the branch mapping and click **OK**.

4. In the **Target** field, specify where you want the source files integrated. The target must be a path that contains some or all of the paths identified in the branch mapping. In other words, the target can be a subset of the path specified by the branch mapping.
5. Specify other options as needed:
 - **Do not copy target files to workspace:** by default, newly branched files are copied to the workspace. To prevent the files from being copied, check this option.
 - **Enable baseless merges:** to perform the integration even if a target file does not share a common ancestor with a source file, check this box.
 - **Disregard previous integration history:** force the integration on all revisions of source and target files, even if some revisions have been integrated in the past. Best used with a revision range.
 - **Propagate source filetypes to target files:** assign the source file's filetype to the target file. If this is not selected, the target file retain its filetype.
 - **Reverse mappings in branch view:** swap source and target mappings when performing this integration.
 - **Enable integrations around deleted revisions:** allow an integration that opens the target file for delete when the source file has been deleted, and allow an integration that branches a new target file that was previously deleted.
 - **Do not get latest revision of selected files:** don't automatically get the latest revision of target files before integrating. Use the current workspace revision instead.
 - **Pending Changelist:** to perform the integration using a changelist other than the default changelist, choose the desired changelist from this drop-down list.
6. To preview the results of the operation, click **Preview**.
7. To integrate the files using the selected branch mapping, click **Merge**.

Managing labels

You can use labels to:

- Mark important file revisions, for example the set of file revisions used to build a particular software release
- Specify groups of related file revisions in operations such as getting file revisions (syncing) and integration

To use labels, you first define the label and then apply the label to file revisions in the depot. When you define a label, you specify a label *view*, which limits the files to which the label can be applied.

Create labels

1. Select **File > New > Label**.
2. In the **Label** form, enter a name for the new label.
3. Enter a description of the label.
4. To prevent other users from modifying the label, select **locked**.
5. Specify the label view one of two ways:
 - **Textually**: on the **Form** tab, in the **View** field, specify the depot location of the files.
 - **Graphically**: on the **View** tab, select depot paths.
6. Click **OK**.

Label files

To label a file, it must be in the label view and synced to your client workspace.

1. Select a file and choose **Actions > Label**.
2. In the **Label Files** dialog, enter or browse for the label you want to use.
3. Click the **Apply label to files** radio button.
4. (Optional) Add files to the label or remove them by clicking the **Add** or **Remove** buttons next to the **Files/Folders** field.
5. Choose whether to label the latest revision of the files or a specific revision.
You can specify a revision by revision number, changelist, date/time, workspace, or other label.
6. Choose whether to exclude deleted revisions of the files.
7. Click **Label**.

To remove a label from a set of files:

1. Select a file and choose **Actions > Label**.
The **Label Files** dialog is displayed.
2. Browse for the label you want to use from the **Label** field.
3. Choose the **Remove selected label from files** option.
4. Click **Label**.

Display and search for labels

To display the labels defined for the Helix Server depot to which you are connected, open the **Labels** tab (click the Labels  icon in the toolbar or go to **View > Labels**).

To search for labels, use the filter fields on the **Labels** tab. You can filter by any combination of the following:

- Owner
- Label name
- File paths

You can enter as many file paths as you like. The filter retrieves labels associated with all files in all file paths entered (restricted by any owner and label name included in the filter). You can use the [File Path Builder](#) to construct file paths.


Note

Global labels display as gray when you are connected to an edge server. Grayed-out labels are not editable when you are connected to the edge server, regardless of permissions, but you can still use them.

For more information on using filters, see "[Searching and filtering](#)" on page 31.

Edit labels

You can edit a label at any time. Editing a label does not change the list of files to which the label is applied.

1. Go to **View > Labels** or click the Labels  icon in the toolbar to open the **Labels** tab.
2. Right-click the label and select **Edit Label 'label_name'**.
The **Label** form is displayed.
3. Enter your changes, and click **Save**.

Delete and unload labels

Deleting a label makes it unavailable for use and removes the label from files. *Unloading* transfers infrequently-used metadata from the versioning engine's database files to a set of flat files in an unload depot. If you unload a label, you can *reload* it if you change your mind and want to use it again.



To delete a label:

1. Go to **View > Labels** or click the Labels  icon in the toolbar to open the **Labels** tab.
2. Right-click the label and select **Delete Label 'label_name'**.

To unload a label:

1. Go to **View > Labels** or click the Labels  icon in the toolbar to open the **Labels** tab.
2. Right-click the label and select **Unload Label 'label_name'**.


To reload an unloaded label:

1. Go to **View > Labels** or click the Labels  icon in the toolbar to open the **Labels** tab.
2. Select the  **Unloaded** icon in the filter pane to open the **Unloaded Labels** dialog, where you can filter for and select unloaded labels to reload.
3. Right-click the label and select **Reload label 'label_name'**.

For more information about unloading, see the [P4 Command Reference](#).

Retrieve file revisions in a label

To retrieve a labeled file revision using the **Get Revision** dialog:

1. Go to **View > Labels** or click the Labels  icon in the toolbar to open the **Labels** tab.
2. Right-click the required label and select **Get Revision**.
The **Get Revision** dialog opens.
3. Click **Add** to open the **Add Files/Folders** dialog.
4. To retrieve all file revisions in a label, click the top-level folder (//). To retrieve a subset of the file revisions in the label, browse to the folders and files that you want and select them.
5. Click **OK** to display the selected folders in the list of files to be retrieved.
6. To ensure that your workspace contains only the labeled file revisions, select **Remove files from workspace if they are not in label**.
7. Click **Get Revision** to retrieve the labeled file revisions.

Display files associated with a label

To display all files associated with a label, do the following:

1. On the **Labels** tab, select a label to display its associated files in a pane at the bottom of the tab.
The pane has its own **Details** and **Files** tabs.

Note

If the pane is collapsed (not visible), move the pointer over the bottom tab border until it changes into up/down arrows; then click and drag it upward to expand the pane.

2. Click the **Files** tab to view the list of files associated with the label.

Managing jobs

Jobs enable you to record requests for work. You can associate jobs with changelists to track the work done to fulfill the request. When the changelist is submitted, the job can be closed. Jobs are displayed on the Jobs tab.

Create a job

1. Open the **Job** form by doing one of the following:
 - Choose **File > New > Job**.
 - Context-click anywhere in the job list pane on the **Jobs** tab.
2. Fill in the **Job** form.

The fields that appear on the Job form depend on the customizations set up by your Helix Server administrator. See "Customizing Perforce: Job Specifications" in the *Helix Versioning Engine Administrator Guide: Fundamentals*.

Add a job to a pending changelist

To add a job to a pending changelist: drag the job from the **Jobs** tab of the right pane to the **Jobs** field of the changelist. You can also use the **Link jobs to changelist** field in the **Submit** dialog. Specify the job status upon submitting the changelist.

You can configure P4V to populate changelists with jobs automatically by setting your P4V user preferences. For more information, see [Editing User Preferences](#).

View jobs

To show the **Jobs** tab, go to **View > Jobs**.

To specify the columns displayed in the Jobs list view: context-click in the column heading and enable or disable the desired columns. **To change the order in which columns are displayed,** drag the column headings right or left to the desired position.

To sort jobs by date: by default, P4V displays dates using the operating system format, which does not sort chronologically. For true chronological sorting, configure P4V to display dates using the Helix Server format of yyyy/mm/dd. Go to **Edit > Preferences** (Windows) or **P4V > Preferences** (Mac) and, on the **Display** page, click the **Format dates using Perforce standard** option.

Search for jobs

Search for jobs by opening the **Jobs** tab and entering filter criteria using standard Helix Server filter expressions (whose syntax is similar to Unix regular expressions) or by using the Job Query Builder to create queries. Search results appear in the window below the query fields and change automatically as you modify your queries.

Click on a job to view details.

Search for jobs using Helix Server filter expressions

If you are comfortable using standard Helix Server filter expressions, enter a search query in the **Keywords or search query** field. You can also click the drop-down arrow to view and select recent queries.

Valid expressions include the following:

Syntax	Description	Example
<i>word word word</i>	Words separated by spaces indicate that the job must contain all of the words in the string in any of the job fields to be included in the filter. Equivalent to the logical operator "and."	filter file mailbox Displays jobs containing <i>all</i> of the words "filter", "file", and "mailbox" in any of the job fields.
pass:[<i>word word word</i>]	Displays jobs that contain any of the specified words. Equivalent to the logical operator "or."	pass:[filter file mailbox] Displays jobs containing the words "filter", "file", or "mailbox."

Syntax	Description	Example
<i>^word</i>	Displays jobs that do not contain the specified word. The 'not' (^) operator cannot be used alone or with the 'or' operator (), only with the 'and' operator (& or space).	<code>filter ^file</code> Displays jobs that contain "filter" and do not contain "file".
<i>fieldname=value</i>	Displays jobs that include the specified value in the specified field.	<code>status=open owner=edk</code> Displays open jobs owned by edk.
<i>^fieldname=value</i>	Displays jobs that do not include the specified value in the specified field. The 'not' (^) operator cannot be used alone or with the 'or' operator (), only with the 'and' operator (& or space).	<code>^status=closed& subsystem=parser</code> Displays unclosed jobs affecting the parser subsystem.

Syntax	Description	Example
<i>fieldname=value+*</i>	Displays jobs that contain the specified value in the specified field, including any combination of characters in the position of the asterisk wildcard.	owner=**ed* Displays jobs in which the value of field "owner" contains the substring "ed," including such values as "Ted," "Edk," and "Fred".
<i>yyyy/mm/dd:hh:mm:ss</i>	Displays jobs that contain the specified date, where "yyyy" is the year expressed in four-digit format and "mm", "dd", "hh", "mm" and "ss" are the month, day, hour, minute and second, respectively, expressed in two-digit format.	2000/02/12:08:30:00 Displays jobs that contain the date February 12, 2000, 8:30 am.

For more information about Helix Server job query syntax, see ["Defect Tracking"](#) in the *Helix Versioning Engine User Guide*.

Search for jobs using file paths

You can view jobs associated with particular files by entering the file path under **Files match any of the following file paths** or by using the **File Path Builder**. The file path you enter retrieves jobs associated with changelists that include any of the files in the path. Any filter criteria that you or the Job Query Builder enter in the **Keywords or search query** field serve to filter the search results further. You can enter either a depot or workspace file path.

To enter file paths directly into the **Files match any of the following file paths** field, do any of the following:

- Use standard Helix Server file path syntax (`//depot/folder/folder/filename` or `//depot/folder/...`). You can use the standard Helix Server wildcards (* and ...) and revision specifiers (@ and #).

For more information on wildcards and revision specifiers, see "[Issuing P4 Commands](#)" in the [Helix Versioning Engine User Guide](#).

- Drag and drop a file path from the Depot or Workspace tree into the field.
- Click the drop-down arrow to view and select recent file paths.

To get help constructing a file path, use the [File Path Builder](#).

Search for jobs using the Job Query Builder

The Job Query Builder helps you to construct filter expressions interactively.

1. Click the **Construct a search query**  icon to open the **Job Query Builder**.

2. Build a filter expression:

■ Select **Match** criteria:

- *All* retrieves results that meet all of the conditions you enter; equivalent to the logical operator "and."

Use *All* to construct more restrictive searches. For example, if you want to retrieve only the jobs that contain both the term "installation" and the term "administration," use *All*.

- *Any* retrieves results that meet any of the conditions you enter; equivalent to the logical operator "or."

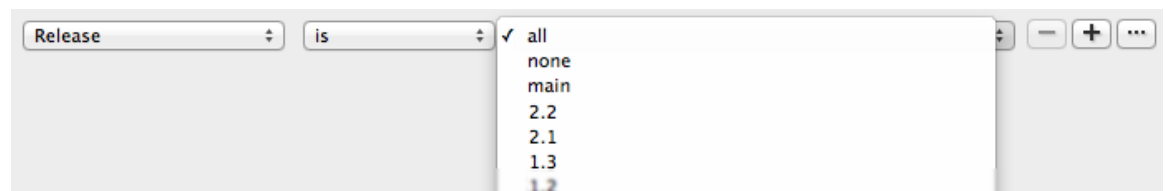
Use *Any* to construct less restrictive searches. For example, if you want to retrieve the jobs that contain at least one of the terms "installation" or "administration," use *Any*.

■ Select a field, operator (*contains, is, does not contain, is empty, etc*) and value:





A screenshot of a filter expression builder interface. It shows a field dropdown set to 'OwnedBy', an operator dropdown set to 'starts with', and a text input field containing 'bruno'. To the right of the input field are three buttons: a minus sign, a plus sign, and a three-dot menu icon.

Some values are editable; others are predefined and can be selected from a drop-down list:



A screenshot of a filter expression builder interface. It shows a field dropdown set to 'Release', an operator dropdown set to 'is', and a dropdown menu open showing a list of predefined values: 'all' (with a checkmark), 'none', 'main', '2.2', '2.1', '1.3', and '1.2'. To the right of the dropdown menu are three buttons: a minus sign, a plus sign, and a three-dot menu icon.

- To add conditions, click the plus  button.
- To remove conditions, click the minus  button.

- To nest conditions (for example, if you want to select any *open* job owned by user *bruno* and containing the term *installation* or *administration*), click the **...** button:

Match **All** of the following conditions:

Status	is	open	-	+	...
OwnedBy	is	bruno	-	+	...

A nested row appears, with the option to match *Any* or *All* conditions:

Match **All** of the following conditions:

Status	is	open	-	+	...
OwnedBy	is	bruno	-	+	...
Any of the following conditions:					
Any Field	contains	installation	-	+	...
Any Field	contains	administration	-	+	...

As you enter or select values, the resulting file path appears in the **Query preview:** field:

Query preview:

```
Status=open OwnedBy=bruno (*installation* | *administration*)
```

3. Click **OK**.

The query that you built appears on the **Jobs** tab in the **Keywords or search query:** field.

7 | Advanced P4V options

This chapter describes options for advanced P4V users and command-line users.

Configuring custom tools	102
Environment considerations	102
Add custom tools	102
Import and export tools	105
Launching P4V components from the command line	106

Configuring custom tools

You can add commands and folders to the Tools menu. The tools can then be accessed from the Tools menu in P4V and optionally from context menus. For each custom tool, you can also create a shortcut. You can export your custom tool definitions to a file from which other P4V users can import them.

Environment considerations

P4V launches the specified custom tool in an external process. Perforce settings are exported to the external process as follows: the **P4PORT**, **P4USER**, **P4CLIENT** and **P4CHARSET** settings in effect for the P4V connection are exported to the external process, plus any other Perforce settings present in the environment in which P4V is running, including command-line arguments specified when P4V was launched and global settings (on Windows machines, settings configured using the **p4 set -s** command).

If the **P4CONFIG** environment variable is set, the tool recognizes any **P4CONFIG** files present in the file hierarchy on which it is operating. Settings in **P4CONFIG** files override settings in the environment. To ensure that **P4CONFIG** settings do not override environment settings, select **Ignore P4CONFIG files**. (For details about config files, refer to the *P4 User's Guide*.) If you are passing arguments to a custom tool, select **Ignore P4CONFIG files** to ensure that the server settings used by the custom tool correspond to the values that are passed.

On a Mac, custom tools are not launched from a shell by default. For tools that must recognize **P4CONFIG** settings, create a script that launches a shell before invoking the tool, or define **P4CONFIG** globally (for example, in **/etc/profile**, **/etc/profile.local**, or **/etc/csh.login**). To spawn a shell using Bourne shell, issue the **#!/bin/sh** command. For the C shell, issue the **#!/bin/tcsh** command.

If you need to keep a tool running or launch another application, run it in the background and exit the shell or process that was launched by the custom tool. This approach ensures that you do not block another tool and that the application will remain open when you exit P4V.

Add custom tools

To add custom tools to P4V:

1. Choose **Tools > Manage Custom Tools**.

The Manage Custom Tools dialog is displayed.

2. Click **New** and choose **Tool....**

The Add Custom Tool dialog is displayed.

3. In the **Name** field, enter the name of the menu item you want to appear on the Tools menu.
4. Use the **Placement** field to define where you want the tool to appear in the Tools menu. Select the **Add to applicable context menus** check box if you want the new tool to be available in P4V context menus.
5. Specify the application that you want to associate with this command in the **Application** field. To browse to the application, click **Browse**. (You can drag and drop the application from an Explorer or Finder window to the **Application** field.)

Important

To define tools that issue p4 commands, specify **p4** in the **Application** field and the command and desired arguments (for example, **fix -c %c job000001**) in the **Arguments** field. In general, be sure to specify the application and arguments separately in the fields provided.

6. In the **Arguments** field, enter command arguments using the specifiers listed in the following table.

Note

Some arguments have an uppercase and lowercase form. To run the tool once, submitting all values together, specify the uppercase argument. To run the tool once for each value, specify the lowercase form.

%a	Selected label
%b	Selected branch
%C, %c	Selected changelists
%D, %d	Selected files or folders (depot syntax used for depot files)
%F, %f	Selected files (workspace syntax used for depot files)
%i	Selected workspace specifications
%J, %j	Selected jobs
%O, %o	<i>Depends on which pane has focus:</i> <ul style="list-style-type: none"> ■ When left pane has focus, selected checked out files. ■ When right pane has focus, all checked out files.
%P, %p	Selected pending changelists
%S, %s	Selected submitted changelists
%t	Selected stream
%u	Selected user
\$c	Current workspace specification
\$p	Current port
\$r	Current client workspace root
\$u	Current user
\$D	Arguments from prompt dialog
%%	Literal '%' character
\$\$	Literal '\$' character

Notes about arguments:

- Arguments must be separated by white space.
- To specify values containing spaces, double-quote them. For example:

```
"C:\Program Files\scripts\changelistreport.pl" %p
```
- You can specify only one **%** argument per tool definition.
- Uppercase specifiers send values to the command as a group, while lowercase specifiers run the command once for each value.
 For example: **mycommand %F** runs **mycommand arg1 arg2 arg3** and **mycommand %f** runs **mycommand arg1; mycommand arg2; mycommand arg3**.
- If your custom tool includes arguments from a prompt dialog (**\$D**), you must warn users to avoid the "&" character when they enter a string value in your custom tool prompt. The "&" value is a special character.

7. In the **Start in** field, specify the directory where you want the command to execute. You can specify the directory using operating system syntax (for example:

C:\tmp, or **/tmp**), or specify **%f**, which uses the directory of the selected file.

Note
 If you specify **%f**, omit any other values. For example, **%f\tmp** is invalid.

8. To execute the application in a terminal window that displays standard input and output, select **Run tool in terminal window**.
9. To enable users to enter arguments after they choose the custom tool menu item, select **Prompt user for arguments** and enter a description of the required arguments in the **Description** field. To enable users to browse when prompted, select **Add file browser to prompt dialog**.
10. To refresh P4V after the tool application finishes running, select **Refresh P4V upon completion**.
11. To save your entries, click **OK**.
12. To add a shortcut for the tool, double-click the **Shortcut** column and enter the desired key combination.
 If a shortcut is reserved or already in use, P4V displays a warning message.
13. To exit the Manage Custom Tools dialog, click **OK**.

Import and export tools

You can export the tools you've defined to an XML file that other P4V users can import. Likewise, you can import tools that other P4V users have created.

To import tools:

1. Choose **Tools > Manage Custom Tools...**
The Manage Custom Tools dialog is displayed.
2. Click **Import tools...**
P4V displays the **Read Custom Tools from a File** dialog.
3. Enter the file name or browse to the file and click **Open**.
The **Import Preview** dialog is displayed.
4. Click **Import**.
The tool definitions are imported.

To export tools:

1. Choose **Tools > Manage Custom Tools...**
The Manage Custom Tools dialog is displayed.
2. Click **Export tools...**
The Export Preview dialog is displayed.
3. Select the tools you want to export and click Export.
The **Save Custom Tools to a File** dialog is displayed.
4. Enter a file name and browse to the location where you want to save the export file and click **OK**.

Note

On Windows, if a tool does not run, verify that the files that it opens have their extensions mapped to the required application.

Launching P4V components from the command line

If you are a P4 command line client user who occasionally uses P4V for certain functions, like viewing the Revision Graph or the Stream Graph, you can use P4VC, a command-line tool included with P4V that enables you to launch discrete P4V components from the command line without having to open a full P4V instance.

P4VC commands use options, arguments, and behavior that match the P4 command line client. For more information about P4VC commands, type **p4vc help** in the command line.

8 | Working with streams

This chapter provides an introduction to Helix Server Streams and describes how to use them.

About streams	107
About stream views	110
Setting up streams	116
Creating new streams	117
Selecting streams	119
Using the stream graph	119
Merging down and copying up between streams	123
Working with task streams	125
Working with virtual streams	131

About streams

Helix Server streams are "branches with brains," a containerized approach to managing bodies of related files such as codelines. Streams associate these bodies of related files with rules that define how you can work with those files, including how you can move changes between them. Most notably, streams are defined hierarchically using the mainline model, and Helix Server generates the views for workspaces that are associated with a stream based on strictly inherited rules. Among the advantages of this approach are (1) change is propagated in a controlled way through the hierarchy that you define, and (2) you can compose the contents of a stream by defining its view, thereby providing all users who work in the stream with a consistent view of its contents.

The mainline model

The mainline model of software configuration management defines a structure based on the stability of the contents of the stream, from softest (unstable or experimental) to firmest (highquality, releasable). Typically, the mainline is required to be fairly table (for example, code must build), development codelines less stable, and release codelines the most stable.

Stream views

A stream is defined not just by its type, but also by its view, which specifies the files and folders that the stream contains and whether they can be edited, merged down, copied up, or branched to a new stream. Stream views define the files that you can work on in your workspace and what you can do with them. They also restrict the files that a child stream inherits from its parent. For example, let's say you have a mainline stream that includes the following directories of files:

```
//Acme/Main
--apps
--api
```

```
--resources  
--docs
```

You want to branch to a development stream. You only need to work in the **apps** folder, but you need resources from the **api** and **resources** folders. You can create a development stream as a child of **Main** that includes the files in the **apps** folder, excludes the files in the **docs** folder, and imports the contents of the **api** and **resources** folders so you can use them but cannot edit and submit any changes to the depot. You assign a stream view to the child stream that enacts these rules. Helix Server generates the workspaces -- and enforces the submit and integration rules -- from the stream view. Any children branched from this new stream inherit those rules. You can assign a more restrictive stream view to subsequent children, but you cannot assign a less restrictive one. Child streams cannot branch more than their parent streams are willing to share.


For more information, see ["About stream views" on page 110](#).

Propagating change between streams

The primary principle of change propagation is **merge down, copy up**. The goal is to keep less stable streams up to date with their more stable parent or child, so that when change is propagated from a less stable stream to a more stable one, no work is overwritten. This approach keeps the resolve operation as simple as possible. Perforce's merge/copy and resolve features are the means by which you propagate change between streams. When you use streams, all changes are propagated between parent and child streams. If you want to merge or copy between peer streams (for example, to merge another developer's changes to your development stream), you must first reparent your stream (that is, edit the stream specification and change the Parent field). This approach is a key benefit of streams: the ability to configure the flow of change.

For more information, see ["Merging down and copying up between streams" on page 123](#).


Stream depots

Streams are stored in a stream depot, which is displayed in P4V like this: . You cannot add a stream to a "classic" Helix Server depot. You can add stream depots through P4Admin or the P4 command-line client.

For more information, see ["Setting up streams" on page 116](#).

Stream types

Perforce codifies the soft-to-firm model of stability in its standard stream types:

-  **Mainline:** A stream with no parent. Expects merging down from more stable child streams. Expects copying up from less stable child streams. Used as the stable trunk of a stream system.

- **Release:** A stream that is more stable than its parent. Expects merging down from more stable child streams. Does not expect copying up from its parent stream. Useful for ongoing stabilization, bug fixing, and release maintenance.
- **Development:** A stream that is less stable than its parent. Expects merging down from its parent stream. Expects copying up from its less stable child streams. Does not expect to have more stable child streams. Useful for long-term projects and major new features.

There are two additional stream types with special characteristics:

- **Task** streams are lightweight, short-term branches that you can use for work that affects a small portion of a full project branch. Task streams enable you to work privately, switch contexts quickly, and reduce the amount of metadata managed by the Perforce service.

For more information, see ["Working with task streams" on page 125](#).

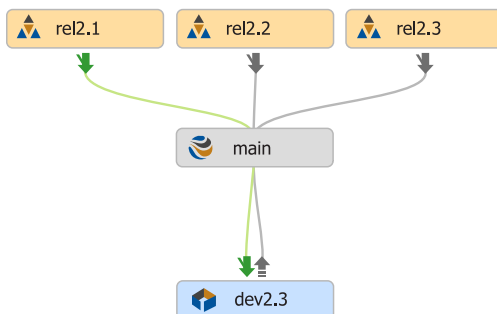
- **Virtual** streams provide users with the ability to restrict the workspace view of a real stream. Virtual streams act as a filter. They are useful when you want to:
 - Submit changes directly to a stream but do not want to sync all of the files in the stream view to your workspace.
 - Develop against the same stream but switch imported libraries, such as when you develop across multiple platforms.

For more information, see ["Working with virtual streams" on page 131](#).

Stream hierarchy in the Stream Graph

In the P4V Streams tab, the Stream graph displays soft streams below the mainline and firm streams above it. For example, the following diagram shows a typical software development structure, consisting of a stable mainline, with experimental streams below it and release-quality streams above:

Figure 1. Basic stream structure for software development



In the stream graph, the most stable streams are always at the top and the least stable at the bottom. The stream graph also displays the parent-child relationships that control the propagation of change between codelines. All streams are children of the mainline, and all children can be parents of yet more children of the same type, enabling specific development and release codelines while ensuring that changes are propagated properly between codelines. The stream graph represents those paths of change propagation using arrow connectors.

For more information, see ["Using the stream graph" on page 119](#).

Terminology differences between command line client and P4V

P4V Term	Command Line Term	Description
Stream Root	Stream	The top level under the depot where files in the stream are stored.
Stream	Name	A descriptive name that you assign and which is displayed to label the streams in the Stream Graph.

About stream views


To configure the files that a stream contains, you define the stream view. The view is composed of a set of paths to which you assign path types that control their behavior. Helix Server uses stream views to generate workspace views and branch views for you.

Mainline, development, and release streams are only visible in the depot tree pane if the stream is populated.

A task stream is only visible in the depot tree pane if that task stream is populated, has a workspace, and that workspace is the selected workspace.

Stream path types

Stream views use the following path types:

- **share:** Files in shared paths can be synced (retrieved), submitted, and integrated (branched, merged, and copied). Shared paths are used for files whose changes will flow to and from other streams.
- **isolate:** Files can be edited but the resulting changes cannot be integrated to other streams. Isolated paths are useful for storing nightly builds and other generated files, like bin directories.
- **import:** Files are sourced from a specified location, and can be synced but cannot be submitted or integrated. Use imported paths for components such as third-party libraries, which are required for building but are never edited. An imported path inherits its view from the parent stream unless its depot location is defined explicitly. Icons of imported files and folders are decorated with a halo  in the parent stream from which they are imported.
- **import+:** Functions like an import path in that it can reference an explicitly defined depot path, but unlike a standard import path, you can submit changes to the files in an import+ path.

- **exclude:** Prevents files that would be inherited from the parent files from becoming part of the stream. Files in excluded paths cannot be synced, submitted, or integrated.

The following table shows the behavior of each path type at a glance:

Path type	Merge/Copy	Branch	Sync	Submit
share	Y	Y	Y	Y
isolate	N	N	Y	Y
import	N	N	Y	N
import+	N	N	Y	Y
exclude	N	N	N	N

Stream path syntax

The syntax for specifying stream paths is similar to Perforce depot path syntax, with the leading `//depot` name and stream name removed. For example:

Depot path: `//Ace/main/qa/...`

Stream view path: `qa/...`

If you want your stream to be able to merge, copy, branch, and submit edits to qa, you'd include a share path like this in the stream view:

```
share qa/...
```

Note

Stream views do not allow positional specifiers (`%%1`) or overlay mapping (`+`).

Helix server rejects leading wildcards. However, it accepts a single trailing asterisk (`*`). For example, if you use `share *` in the `Paths:` field in the stream spec, Helix server accepts it and interprets it as a wildcard for the current directory.

Inheritance between parents and children

Child streams inherit folder paths and behavioral rules from their parents. When we talk about inheritance between parents and children, it helps to think in the following terms:

- *Permissiveness*: what actions (submit, sync, etc) are permitted on a path?

Path types are inherited from parent streams, and you cannot override the effects of the path types assigned by parent streams. In other words, child streams are always as permissive or less permissive than their parents, but never more permissive. For example, if a parent stream defines a path as **isolate**, its child streams cannot redefine the path as **share** to enable integrations.

- *Inclusiveness*: what paths are included in the stream?

Since children cannot, by definition, be more inclusive than their parents, you cannot include a folder path in a child that is not also included in its parent. That means, for example, that you cannot add an **isolate** path to a child if the folders in that path are not also included in the parent. In the following example, the incorrectly defined Dev stream, which is a child of **Main**, contains an **isolate** path that will not work, because it includes folders that are not included in the parent. In order to isolate the config/ folder in the Dev stream, that folder has to be included as a **share** or **isolate** path in **Main**:

Incorrect	Correct
Stream: //Acme/Main	Stream: //Acme/Main
Parent: none	Parent: none
Paths: share apps/...	Paths: share apps/...
share tests/...	share tests/...
	share config/...
Stream: //Acme/Dev	Stream: //Acme/Dev
Parent: //Acme/Main	Parent: //Acme/Main
Paths: share apps/...	Paths: share apps/...
share tests/...	share tests/...
isolate config/...	isolate config/...

Examples

This section includes simple and more complex examples of stream usage.

Simple share

Let's start with a simple case: two streams, **//Ace/main** and its child, **//Ace/dev**.

```
Stream: //Ace/main
Parent: none
Paths: share ...
Stream: //Ace/dev
Parent: //Ace/main
Paths: share ...
```


In this case, the entire stream path is shared. When you switch your workspace to the `//Ace/main` stream, the workspace view looks like this:

```
//Ace/main/... //your_ws/...
```

The workspace view maps the root of the `//Ace/main` stream to your workspace. When you switch your workspace to the `//Ace/dev` stream, the workspace view is this:

```
//Ace/dev/... //your_ws/...
```

And the branch view for `//Ace/dev/` looks like this:

```
//Ace/dev/... //Ace/main/...
```

In other words, the entire dev stream can be synced to workspaces, and the entire stream can be branched, merged, and copied.

Share and import

Let's look at an example where software components are housed in three separate depots, `//Acme`, `//Red`, and `//Tango`.

The Acme mainline is configured like this:

```
Stream: //Acme/Main
Parent: none
Paths: share apps/...
       share tests/...
       import stuff/... //Red/R6.1/stuff/...
       import tools/... //Tango/tools/...
```

If you switch your workspace to the `//Acme/Main` stream, this would be your workspace view:

```
//Acme/Main/apps/... //your_ws/apps/...
//Acme/Main/tests/... //your_ws/tests/...
//Red/R6.1/stuff/... //your_ws/stuff/...
//Tango/tools/... //your_ws/tools/...
```

The stream's Paths field lists folders relative to the root of the stream. Those are the folders you will get in your workspace, beneath your workspace root. The shared folders are mapped to the `//Acme/Main` path, and the imported paths are mapped to their locations in the `//Red` and `//Tango` depots.

Share, isolate, exclude, and import

Let's say that your team doesn't want to do actual development in the mainline. In this example, the XProd feature team has a development stream of their own, defined like this:

```
Stream: //Acme/XProd
Parent: //Acme/Main
```

```

Paths: import ...
       isolate apps/bin/...
       share apps/xp/...
       exclude tests/...

```

Switching your workspace to the **//Acme/XProd** stream gives you this view:

```

//Acme/Main/apps/... //your_ws/apps/...
//Acme/XProd/apps/bin/... //your_ws/apps/bin/...
//Acme/XProd/apps/xp/... //your_ws/apps/xp/...
//Red/R6.1/stuff/... //your_ws/stuff/...
//Tango/tools/... //your_ws/tools/...
-//Acme/XProd/tests/... //your_ws/tests/...

```

Here we see stream view inheritance at work. Imported paths are mapped to whatever they're mapped to in the parent's client view. The shared and isolated paths are mapped to the child stream; these contain the files the XProd team are working on and will be submitting changes to. And the excluded path (marked with a minus sign in the view) doesn't appear in the workspace at all.

Because the **//Acme/XProd** stream has a parent, it has a branch mapping that can be used by the copy and merge commands. That branch view consists of the following, with just one path shared by the child and parent (note that you must use P4, the Perforce Command Line Client, to view stream branch views):

```

-//Acme/XProd/apps/... //Acme/Main/apps/...
-//Acme/XProd/apps/bin/... //Acme/Main/apps/bin/...
//Acme/XProd/apps/xp/... //Acme/Main/apps/xp/...
-//Acme/XProd/stuff/... //Acme/Main/stuff/...
-//Acme/XProd/tests/... //Acme/Main/tests/...
-//Acme/XProd/tools/... //Acme/Main/tools/...

```

When you work in an **//Acme/XProd** workspace, it feels as if you're working in a full branch of **//Acme/Main**, but the actual branch is quite small.

Child that shares all of the above parent

Let's suppose that Bob, for example, creates a child stream from **//Acme/XProd**. His stream spec looks like this:

```

Stream: //Acme/BobDev
Parent: //Acme/XProd
Paths: share ...

```

Bob's stream has the default view template. Given that Bob's entire stream path is set to "share," you might expect that his entire workspace will be mapped to his stream. But it is not, because inherited behaviors always take precedence; sharing applies only to paths that are shared in the parent as well. A workspace for Bob's stream, with its default view template, will have this client view:

```
//Acme/Main/apps/... //your_ws/apps/...
-//Acme/BobDev/tests/... //your_ws/tests/...
//Acme/BobDev/apps/bin/... //your_ws/apps/bin/...
//Acme/BobDev/apps/xp/... //your_ws/apps/xp/...
//Red/R6.1/stuff/... //your_ws/stuff/...
//Tango/tools/... //your_ws/tools/...
```

A workspace in Bob's stream is the same as a workspace in the XProd stream, with one exception: the paths available for submit are rooted in **//Acme/BobDev**. This makes sense; if you work in Bob's stream, you expect to submit changes to his stream.

By contrast, the branch view that maps the **//Acme/BobDev** stream to its parent maps only the path that is designated as shared in both streams:

```
-//Acme/Main/apps/... //XProd/apps/...
-//Acme/BobDev/tests/... //XProd/tests/...
-//Acme/BobDev/apps/bin/... //XProd/apps/bin/...
//Acme/BobDev/apps/xp/... //your_ws/apps/xp/...
-//Red/R6.1/stuff/... //XProd/stuff/...
-//Tango/tools/... //XProd/tools/...
```

The default template allows Bob to branch his own versions of the paths his team is working on, and have a workspace with the identical view of nonbranched files that he would have in the parent stream.

Remap files

Files can be remapped, which enables you to place them in a workspace location that differs from their depot location. For example, the following view specifies that the parent's **docs** files reside beneath the **doctools** path in workspaces defined for this stream.

Remapped:

```
Remapped:
docs/... doctools/docs/...
```

Ignore files

You can specify file types to be ignored, which is useful for ensuring that artifacts such as object files or other interim files are never checked in or integrated. These types are excluded by the workspace view of workspaces associated with the stream. For example:

Ignored:

```
.swp  
.o  
/tmp/...
```

Setting up streams

To set up streams for the first time, you must create a stream depot and create and populate a mainline stream.

To create a stream depot:

1. Launch P4Admin (from P4V, choose **Tools > Administration**) and connect to the Perforce service as a user that has admin privilege.
2. Choose **File > New... > Depot**, specify a name for the stream depot, and click **OK**.
The Depot dialog is displayed.
3. For Depot Type, choose stream.
4. Click **OK**, and verify that your new stream depot is listed in the Depots tab.

To create the mainline stream:

1. Launch P4V and connect to the Perforce service where you want your streams to reside.
2. Choose **File > New > Stream...**
The **Stream** dialog is displayed.
3. Specify the name ("main" or "mainline" is a good descriptive choice), depot, and root folder. Omit parent (the mainline has none). For stream type, choose "mainline".
4. Click **OK** to save your stream specification.
5. Verify that your new mainline stream is displayed in the Streams tab. Note that the mainline stream is not displayed in the Depot tab until you populate it with files.

For more information, see ["Creating new streams" on the facing page](#).

To populate your mainline stream:

1. Copy the desired files and folders to the root folder of the workspace you just created (the mainline stream).
2. In P4V, switch to the left-pane Workspace tab. Your files are displayed undecorated (that is, as plain file icons), which indicates that they are not under Helix Server control.
3. Context-click the top-level folder and choose **Mark for Add...**
P4V displays the **Select Pending Changelist** dialog.

4. Click **OK**.

In the Pending Changelists tab, verify that your files are listed in the default changelist, decorated with a red plus sign.

5. Context-click the default changelist and choose **Submit...**

The **Submit Changelist** dialog is displayed.

6. Enter a descriptive comment and click Submit.

Note

As an alternative, you can populate streams by integrating files from existing depot locations.

To verify that your files have been submitted, check the Log pane at the bottom of the screen and the left-pane Depot tab. If your submission succeeded, your files are listed in the mainline folder in your stream depot. Now you can branch the files in the mainline to development and release streams, edit files in those streams, and propagate changes among them.

Creating new streams

To create a stream:

1. Choose **File > New > Stream** or right-click a stream in the [Stream Graph](#) and select **Create New Stream from '<stream name>'...**

Note

Streams must be created in a Streams depot. You cannot create a stream in a classic depot.

2. In the **Stream** dialog, specify the name of the stream.

By default, the name you specify is used as the name of the folder (under the stream depot folder) in the depot where files in the stream are stored. To override the default folder name, go to the **Advanced** tab.

3. Choose the stream type.
4. (Optional) If you require a flow of change that is different from the default for the stream type, enable or disable the corresponding **Change propagation** option (**To parent** and **From parent**).

- For release and development streams, choose the parent stream; for mainline streams, enter the depot where the mainline will be stored.

By default, the stream inherits its parent's stream view. To edit the view of the child stream, go to the **Advanced** tab.

The Advanced tab options enable you to do the following:

- Edit the inherited view.
- Change the default locations of files mapped from the depot.
- Lock the stream settings so that only the owner can edit them.
- Restrict check-ins to the stream owner.
- Allow to merge the stream's content both up and down.
- Specify file or folder patterns that you want ignored by the stream.

Ignoring a file or folder pattern prevents users from submitting files or folders that follow that pattern, no matter the stream view definition. You can use Ignore to ensure that generated files are never checked in, for example. You can enter filenames, filename extensions, or file paths. Separate each with a line break:

```
.swp .o /tmp/...
```

Note

Stream specifications do not allow wildcards (*), positional specifiers (%%1), or overlay mapping (+).

Mainline, development, and release streams are only visible in the depot tree pane if the stream is populated.

- (Optional) Select whether to **Create a workspace for use with this stream..**
If you choose not to [create a workspace](#) immediately, you can do so later, when you want to work in the stream.
- (Optional) Select whether to **Branch files from parent on stream creation.**
If you choose not to branch files immediately, you can do so later using the Copy operation.
- To save your entries, click **OK**.

Creating stream workspaces

To work in a stream, you must have a workspace for it. Helix Server generates the view (mapping of depot locations to client locations) for any stream workspace, but you must initiate the creation of the workspace. You can do so in the following ways:

- When you create a new stream using the **Stream** dialog, select the **Create a workspace for use with this stream** option.

- The first time that you select a stream to work in, you'll be prompted to create a new workspace. See "[Selecting streams](#)" below.

The [Workspace dialog](#) opens, with all necessary fields pre-populated to work with the current stream.

You can update any of the fields in the Workspace dialog, except Stream name and Workspace mappings (which can only be changed by updating the stream's path configuration). If you want a "back-in-time" view of your stream, you can use the **Stream At Change** option to sync the stream using the stream's view as of a specific changelist. When you use the **Stream At Change** option, you cannot submit changes to the files in the stream, since your workspace files are not at the head revision.

Selecting streams

You can select a stream to work in by doing either of the following:



- Context-click the stream in the Tree Pane or Stream Graph and select **Work in this stream**.
- Double-click the stream in the Stream Graph and select **Work in this stream**.

If you haven't worked in this stream before, P4V prompts you to create a new workspace. You can also add the stream to an existing workspace, if you have enabled this functionality in Preferences. For more information, see "[Work in a stream](#)" on page 122.

Using the stream graph

The Streams tab lists streams or displays a Stream Graph that shows the relationships between parent and child streams in a selected depot.

Display streams

1. Go to **View > Streams** or click the Streams  toolbar button to open the **Streams** tab.
2. Select a display option by clicking the **Show streams in a graph, list or tree**  icon:
 - **List** displays the streams in a flat list.
 - **Tree** displays the streams in a hierarchy, with development streams and release streams listed under mainline streams.
 - **Graph** displays the streams graphically, with stream type, relationship, and merge/copy indicators.

3. Search for streams or restrict the streams that are displayed by using a filter:
 - For lists and trees, filter using the Filter pane at the top of the tab. You can filter by owner, stream name, parent stream, stream type, and depot.
 - For the Stream Graph, filter using the ["Configure the stream graph display" on the facing page](#).
- For more information about filters, see ["Searching and filtering" on page 31](#).

Stream graph display conventions

In the Streams tab, the Stream Graph provides a graphical view of stream relationships and provides tools and shortcuts for working with streams.

The graph uses location and color to depict stream types: mainline streams are gray and placed in the middle of the graph, release streams are orange and appear above the mainline, and development streams are blue and appear below.



Status indicators between streams tell you which streams have changes to contribute and where the changes can be copied or merged:

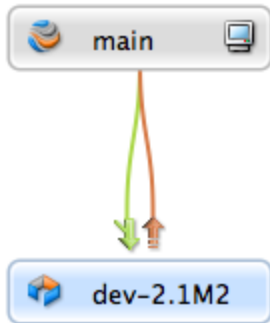
Merge indicator: 

Copy indicator: 

The arrows are color-coded to indicate status:

- **Gray:** no merge or copy required
- **Green:** a merge or copy operation is available
- **Orange:** stream must be updated, after which merge or copy is available


For example, the following arrows above the **dev-2.1M2** stream indicate that you must update it by merging down from its parent, after which you can copy up changes to the parent.



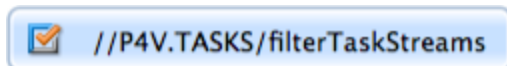
The workspace icon  indicates the stream you are currently working in.

You can drag the workspace icon to another stream to switch your workspace to that stream. The Stream field value in the workspace definition changes to the new stream.

Configure the stream graph display

1. In the Stream Graph view, click the filter  icon on the Streams tab toolbar.
The **Graph View Options** dialog is displayed.
2. Choose the depot containing the streams you want to view from the **Depot** drop-down list.
3. To select the streams you want displayed in the graph, click **Select Streams** and choose the display option, or check the individual streams that you want displayed in the graph.
You may need to expand the tree within the dialog pane to view the streams you want to select.
4. To view the actual depot location of the selected streams in the Stream Graph, select **Stream Root** as your **Display stream** option.


The full depot path appears in the stream node for each stream:



This is useful if you have parent streams with child task streams that are located in other depots. To change the size of the stream node to accommodate long depot paths, you can move the **Stream node width** slider. You can also select these display preferences in **P4V > Preferences > Streams** (Mac) or **Edit > Preferences > Streams** (Windows).

5. Click **Apply**.

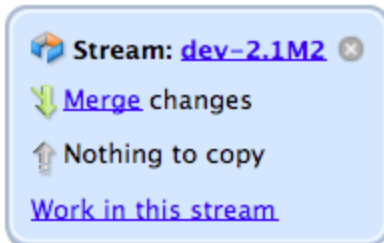
The Stream Graph displays the streams that you specified.

- (Optional) Click the **Graph Navigator**  icon or button to open the Graph Navigator dialog, where you can configure the size of the Stream Graph display and select which portion of the Stream Graph to view.

Use your mouse or cursor keys on the navigator pane to select the portion of the image you want to view.

Display stream status


Double-click a stream to view a popup that contains status details:



Work in a stream

To work in a stream or switch from one stream to another, do one of the following:

- Double-click the stream and select **Work in this stream**.

A warning dialog pops up and asks you to switch workspaces or create a new one. Click the **Switch Workspaces** button to switch your workspace. If more than one workspace is associated with the stream, the **Select Workspace dialog** opens, where you can search for and select the workspace you want. The workspace  icon appears on the stream you are now working in.

Note

If you have set your stream operations preference in the [Preferences](#) dialog to use the same workspace when you switch between streams — or you are using P4Sandbox — then P4V does not prompt you to switch workspaces. Instead the workspace view changes to include the stream you are switching to.

- Drag the workspace icon from the stream you are currently working in to another stream. When you switch streams this way, the workspace view changes to include the stream you are switching to, no matter what you set as your stream operations preference.

Branching with Streams

To create a new development or release branch, you create a child stream from a parent. You can also create virtual siblings of your mainline stream by branching.

Note

You can create task streams by creating a child stream from a parent, but you can also create parentless task streams. For more information about how branch a task stream, see [Working with Task Streams](#).

To create a child stream:

1. In the Streams tab, context-click the stream and select **Create New Stream from 'stream_name'...**
2. In the **Stream:New** dialog, define the new stream.
See "[Creating new streams](#)" on page 117.
3. Verify that the new stream appears correctly in the stream graph.

If you have specified the stream type correctly, more stable streams are displayed above the parent and less stable streams below the parent.

You can also *reparent* streams that have already been branched. To reparent a stream, do one of the following:

- In the Stream graph, drag the stream to the new parent stream.
- In the Streams tab (graph, list, or tree view), context-click the stream, select **Edit Stream 'stream_name'...**, and enter a new parent in the Stream dialog.

Note

You cannot reparent task streams.

Merging down and copying up between streams

Before changes made in a less stable stream can be copied up to its more stable child or parent, any changes in the more stable stream must be merged down to the less stable.

In the Stream graph, status indicators between streams tell you which streams have changes to contribute and where the changes can be copied or merged:

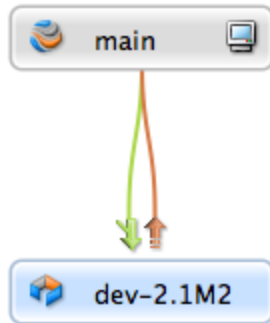
Merge indicator: 

Copy indicator: 

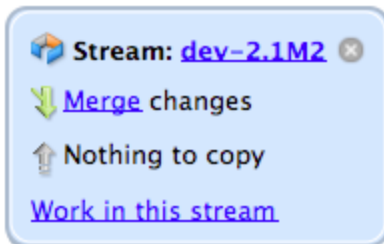
The arrows are color-coded to indicate status:

- **Gray:** no merge or copy required
- **Green:** a merge or copy operation is available
- **Orange:** stream must be updated, after which merge or copy is available

For example, the following arrows above the **dev-2.1M2** stream indicate that you must update it by merging down from its parent, after which you can copy up changes to the parent.



When you double-click a stream in the Stream graph, a pop-up displays copy and merge information, along with links to the Merge/Integrate and Copy dialogs.



Merge down

To merge changes down to a less stable stream:

1. In the Streams tab, context-click the target stream and choose **Merge/Integrate to target stream...** or double-click and choose **Merge changes**.

When you merge down or copy up, you must be working in the target stream.

2. If prompted, select or create a workspace for the target stream.

The **Merge/Integrate** dialog is displayed. Verify that the correct source and target streams are specified.

For more information about the Merge/Integrate dialog, see "[Merging files between codelines](#)" on page 83.

3. (Optional) To specify how the merge is to be resolved, click **Resolve and Submit**.

To enable specific p4 resolve flags, click **Advanced**.

4. Click **Merge**.

If necessary, resolve the merges manually, then submit the resulting changelist.

Copy up

When you copy changes up to a more stable stream, you are propagating a duplicate of the less stable stream.

To copy changes up to a more stable stream:

1. In the Streams tab, context-click the target stream, or double-click the stream and select **Copy changes**.
If prompted, select a workspace for the target stream.
When you merge down or copy up, you must be working in the target stream.
2. In the **Copy** dialog, review the propagation information or refine the operation using the options on the Filter, Submit, and Advanced tabs.
3. To propagate changes to the more stable stream, click **Copy** and submit the resulting changelist.

Propagate change between unrelated streams

To propagate change between streams that are not directly connected, click the **Browse** button on the **Merge** or **Copy** dialog, then click the **Display all streams** link and choose the desired source.

You can also *reparent* a stream to create the relationship. To reparent a stream in the Stream graph, drag the stream to the new parent stream.

Note

You cannot merge or copy changes to unrelated task streams.

Working with task streams

Task streams are lightweight branches that you can use for work that affects a small portion of a full project branch. Task streams enable you to work privately, switch contexts quickly, and reduce the amount of metadata managed by the Perforce service. They are ideal for developing small-to-medium features and bug fixes.

Overview

Task streams are sparse and semi-private:

- When you first create a task stream, only the workspaces associated with that task stream can see the files in the stream.
- When you submit changes to the task stream, the *changed* files are visible to users who are not using the task stream, but that is all they can see.

The workspaces associated with the task stream remain the only ones that can see all files and activity (such as checkouts and shelves).

A task stream is only visible in the depot tree pane if that task stream is populated, has a workspace, and that workspace is the selected workspace.

The typical task stream workflow usually ends with the task stream being deleted or unloaded:

1. [Create a task stream](#) from a parent stream (although a parent is not required).
2. Populate the task stream.
3. Work on a new feature and submit the code.
4. Merge down changes from the parent stream to the task stream.
5. Copy your changes up.
6. [Delete or unload the task stream](#).


You do not always have to delete or unload the task stream. For example, if you find yourself editing more than half the files in your task stream, that stream will no longer provide any savings in server overhead or metadata management activity. It can make sense then to convert your task stream into a regular stream.

Using a task stream feels just like using a regular stream, with the following exceptions:

- Task streams do not require a parent stream.
This enables users who are not working in a stream depot to create task streams. The task stream must reside in a stream depot, but that depot can be no more than a holding place for your task stream. Contact your Helix Server administrator or project lead for information about the stream depot to use. For more information, see "[Create a task stream without a parent](#)" on page 128.
- The parent can reside in another depot.
Task streams can quickly accumulate in a depot until they are deleted or unloaded. To keep a project depot uncluttered by task streams, your Helix Server administrator or project lead may choose to establish certain streams depots as dedicated holding places for task streams. In that case, you create your stream in the task streams depot as a child of a parent in the project depot. Even though the task stream resides in another depot, you can see it displayed in the Stream Graph view of the parent's depot. For more information, see "[Create a task stream in a different depot](#)" on the facing page.
- Task streams cannot have child streams.
- Task streams cannot be reparented.
To reparent a task stream, you must first [convert it into a regular stream](#).

Create a task stream from a parent stream in the same depot

1. In the Streams tab, context-click the parent stream and select **Create New Stream from *'stream_name'*...**

2. In the **Stream:New** dialog, define the new stream.
 - Give the new stream a unique name.
You cannot re-use task stream names, even if the task stream has been deleted.
 - On the **Basic Settings** tab, select a Stream type of **task**.
 - Verify that the depot where the task stream will be located is the current depot and that the parent stream is correct.
See "[Creating new streams](#)" on page 117 for more information about the Stream:New dialog.
3. Verify that the new task stream appears correctly in the Stream Graph.
Refresh the Stream Graph view. You may need to select the task stream under Graph View Options. The task stream should appear as a child of its parent and be decorated with the task stream icon: .

Create a task stream in a different depot

1. Open the **Stream:New** dialog.
You can open the dialog by doing any of the following:
 - Go to **File > New > Stream**
 - In the Streams tab, context-click an existing stream and select **Create New Stream from 'stream_name'**.
 - In the Streams tab, context-click on any white space in the Stream Graph and select **New Stream**.
2. In the **Stream:New** dialog, define the new stream.
 - Give the new stream a unique name. You cannot re-use task stream names, even if the task stream has been deleted.
 - On the **Basic Settings** tab, select a Stream type of **task**.
 - Select the depot where the task stream will be located.
 - Select the parent stream (in a different depot) from which the task stream will be branched.
See "[Creating new streams](#)" on page 117 for more information about the Stream:New dialog.

3. Refresh the Stream Graph view to verify that the new task stream appears correctly.

- In the child stream's depot, the parent stream and parent stream's depot appear grayed out under **Graph View Options**, and the task stream appears as normal.

When you select the child stream for viewing in the Stream Graph, it should appear as a child within its parent stream and parent depot hierarchy and be decorated with the task stream icon:

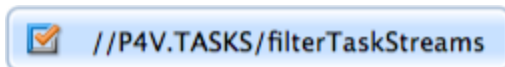
 .

- In the parent's stream depot, the task stream appears grayed out under **Graph View Options**, but you can still select it for viewing in the Stream Graph.

The task stream should appear as a child of its parent and be decorated with the task

stream icon:  .

- When you are viewing the parent stream's depot, you may want to see the actual depot location of the child task stream in the Stream Graph:
 - Select **Stream Root** as your **Display stream** option under Graph View Options. The full depot path appears in the stream node for each stream:



- To change the size of the stream node to accommodate long depot paths, you can move the **Stream node width** slider.

You can also select these display preferences in **P4V > Preferences > Streams** (Mac) or **Edit > Preferences > Streams** (Windows).

Create a task stream without a parent

1. Open the **Stream:New** dialog.

You can open the dialog by doing any of the following:

- Go to **File > New > Stream**.
- In the Streams tab, select the stream depot that will hold the new task stream, context-click an existing stream, and select **Create New Stream from '*stream_name*'...**
- In the Streams tab, select the stream depot that will hold the new task stream, context-click on any white space in the Stream Graph, and select **New Stream**.

2. In the **Stream:New** dialog, define the new stream.
 - Give the new stream a unique name.

You cannot re-use task stream names, even if the task stream has been deleted.
 - On the **Basic Settings** tab, select a Stream type of **task**.
 - Select the depot where the task stream will be located.

For a parentless stream this will usually be a stream depot that is dedicated to holding task streams. Contact your Helix Server administrator or project lead for information about the stream depot to use.
 - Leave the parent stream field blank.

See ["Creating new streams" on page 117](#) for more information about the Stream:New dialog.
3. Verify that the new task stream appears on the Streams tab.

Parentless task streams do not appear in the Stream Graph, since there is no relationship to display. You can view parentless task streams in the Streams tab using List or Tree view.
4. Populate the new task stream.
 - Open the **Branch** dialog by context-clicking the task stream in List or Tree view on the Streams tab and selecting **Branch files....**
 - Whether you **Specify source and target files** or **Use Branch Mapping**, select the files that you want to branch to your new task stream as your source and the new stream as your target.

A parentless stream will not appear in the browse dialog for the Choose target files/folders field. If you opened the Branch dialog by context-clicking the new task stream, it will appear in this field by default. Otherwise, you must enter the full depot path manually.
 - If you didn't do so as part of the branching process, submit the changelist containing the branched files.

For more information about the Branch dialog, see ["Creating branches" on page 82](#).

You can also populate your new task stream using the Merge/Integrate dialog. For more information, see ["Merging files between codelines" on page 83](#).
5. Verify that the new task stream is populated.
 - When you are working in the new task stream workspace, the files should appear in the Depot view.
 - When you get the latest revisions (sync) to your new task stream workspace, the files should appear in the Workspace view.

Convert a task stream to a regular stream

Important

Once you convert a task stream to a regular stream, you cannot convert it back.

To convert a task stream to a regular stream:

1. Context-click the task stream in the Streams tab and select **Edit Stream 'stream_name'**.
2. Change the Stream type to the regular stream type you want.
 - Task streams with parents can convert only to release or development streams.
 - A parentless task stream can convert only to a mainline stream.
 - You cannot convert a task stream to a virtual stream.
3. Verify that the parent stream (if there is one) is in the same depot as the task stream to be converted.

Delete and unload task streams

You should always delete or unload a task stream after you are done with it. Your Helix Server administrator may also run a batch process to delete or unload inactive task streams on a regular basis. Note that any submitted files remain in the depot even after you delete a task stream. Note also that you cannot re-use the names of deleted task streams.

To delete a task stream:


1. Context-click the stream in the Streams tab (in graph, tree, or list view)
2. Select **Delete Stream 'stream_name'**.

Unloading transfers infrequently-used metadata from the versioning engine's database files to a set of flat files in an unload depot. If you unload a task stream, you can reload it if you change your mind and want to use it again.

To unload a task stream:

1. Context-click the stream in the Streams tab (in graph, tree, or list view).
2. Select **Unload stream 'stream_name'**.

To reload an unloaded task stream:

1. Display the Streams tab in List or Tree view.
2. Select the  **Unloaded...** icon in the filter pane to open the Unloaded Streams dialog, where you can filter for and select unloaded streams to reload.
3. Context-click the stream and select **Reload Stream**.

For more information about unloading, see the [P4 Command Reference](#).

Filter task stream files out of File History results

To filter out file revisions that were submitted to task streams when you view file history:

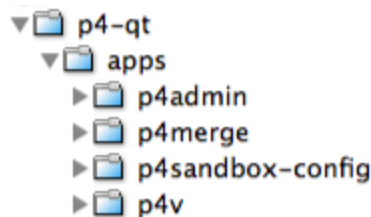
1. Go to **P4V > Preferences**(Mac) or **Edit > Preferences** (Windows) and select **Files and History**.
2. Select **Hide files/revisions from 'task' streams (when following branch, copy actions)**.

Working with virtual streams

Virtual streams provide users with the ability to restrict the workspace view of a real stream. Virtual streams act as a filter. They are useful when you want to:

- Submit changes directly to a stream but do not want to sync all of the files in the stream view to your workspace.
- Develop against the same stream but switch imported libraries, such as when you develop across multiple platforms.

For example, let's say you are working in a development stream whose view includes all of the files in the *apps* directory:



If you want to work only on the contents of the *p4admin* folder, you might not want the burden of syncing all of the files in the *apps* directory to your workspace. You cannot change the workspace view of the development stream to exclude the folders you do not want; the workspace view is inherent to the stream. You could create a child of the development stream that excludes the files you do not want, but then you would be unable to submit your changes directly to the development stream; you would be subject to an unnecessary merge-down/copy-up routine every time you submit. Instead, you create a virtual stream as a child of the development stream that excludes all but the *p4admin* folder:

1. In the [Stream editor](#), you select a stream type of *Virtual*.
2. In the Paths field, you enter stream paths that exclude all but the *p4admin* folder:

```

exclude ...
share p4-qt/apps/p4admin/...

```

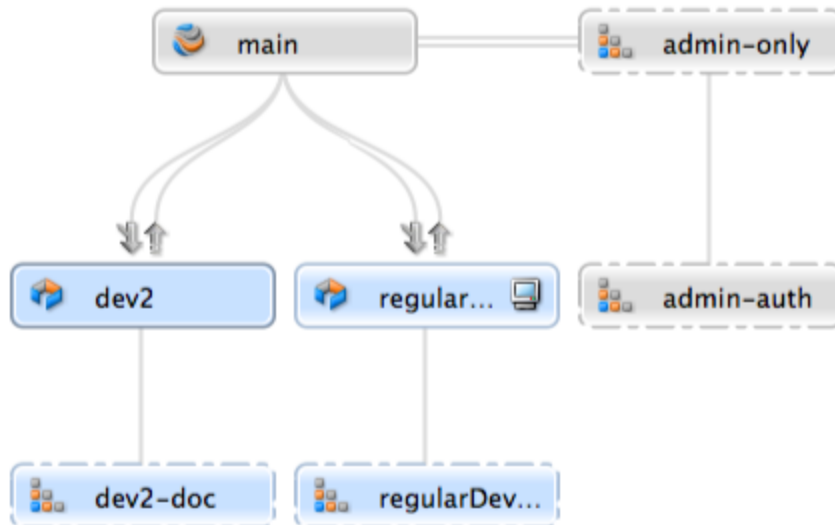
This gives you a virtual stream with the workspace view you want:

```
//depot/development/p4-qt/apps/p4admin/... p4-qt/apps/p4admin/...
```

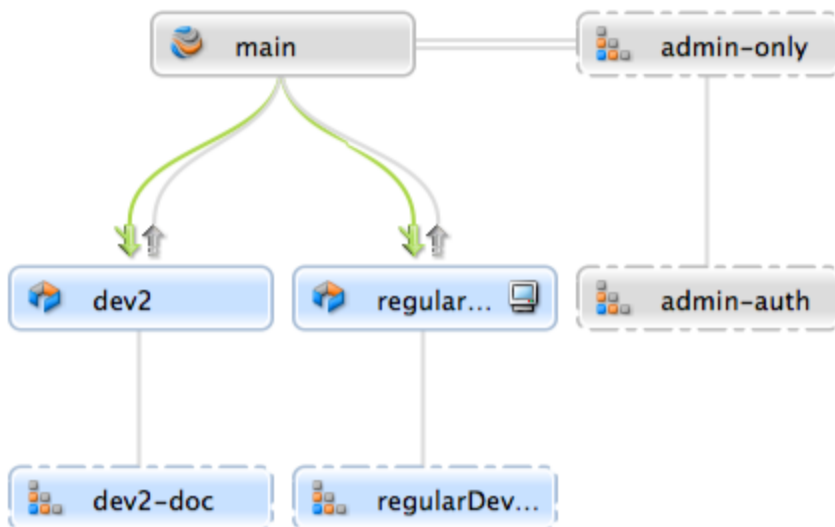
When you submit changes to the virtual stream, those changes are submitted directly to its parent development stream.

View virtual streams

In the Streams Graph, a virtual stream is represented with a dashed border, and its relationship with its base parent stream is represented by a gray line without merge or copy indicators. Virtual children of the mainline are displayed next to the mainline. All other virtual streams are displayed above or below their base parents, depending on whether their parents are development or release streams:



The streams graph shows no merge or copy indicators for virtual streams. Rather, the base parent of a virtual stream carries the merge and copy indicators, because the *actual* merges and copies flow between real streams. When you submit a change through a virtual stream, the copy up or merge down arrows appear between the virtual stream's base parent and the real streams that should be merged down or copied up to. For example, if you submit a change to *main* through the *admin-auth* virtual stream, merge arrows appear between the *main* stream and its real children:



Stream path behavior in virtual streams

You use stream paths to define a virtual stream just as you do when you define a real stream. However, the path types can behave differently. In particular, **isolate** and **import** can behave like **share** in the context of virtual streams and may therefore be redundant when you define your virtual stream view:

- **Share:** in real streams, files included in **share** paths can be merged, copied, branched, synced (retrieved) and submitted to the depot. In virtual streams, files included in **share** paths cannot be submitted, merged, copied, or branched, since, by definition, virtual streams do not allow those actions (submitting, branching, merging, and copying actually happen through the virtual stream's real parent).
- **Isolate:** in real streams, files included in **isolate** paths can be synced to your workspace and submitted to the depot, but not branched, merged, or copied. In virtual streams, **isolate** functions essentially like **share**, since virtual streams do not, by definition, allow submits, branches, merges, or copies. This does not mean that there is no reason to use **isolate** when creating a virtual stream, however; if you intend to create a real child from the virtual stream, the files in the **isolate** path in the real child stream will behave according to expected **isolate** behavior.
- **Import:** in real streams, files included in **import** paths can be synced but not merged, copied, branched, or submitted. In virtual streams, **import** behaves like **share** if your **import** paths are importing from the real parent. However, if you import an explicit depot path from another location, then typical **import** behavior occurs.

For example, let's say your parent stream, `//Acme/Main`, includes the following file directory:

```
//Acme/Main/lib
```

When you define a virtual stream as a child of `//Acme/Main`, the following import path is redundant, since it behaves just like a **share**:

```
share ...
  import lib/...
```

The **import** path is superfluous, unless you intend to create real children from the virtual stream. However, if you want to import libraries from an explicit depot location to your virtual stream, your **import** path is necessary and functions just as it would for a real stream.

```
share ...
  import lib/... //Red/R6.1/stuff/...
```

- **Exclude:** always behaves just as it does in real streams.

For more information about stream paths, see the section "Stream path" in the "Streams" chapter of the [Helix Versioning Engine User Guide](#).

Create a virtual stream

You can create a virtual stream as a child of any stream type — main, development, release, or another virtual stream. The virtual stream takes the behavior of its parent stream type (that is, a virtual stream created as a child of a main stream acts like a main stream, and a virtual stream created as a child of a development stream acts like a development stream).

1. In the **Stream Graph**, context-click a stream and select **Create New Stream from '<stream name>'...**
2. On the **Basic Settings** tab in the **Stream** dialog, select a **Stream type** of **virtual - used to narrow the scope and submit directly to the parent.**
3. On the **Advanced** tab in the **Stream** dialog, enter the stream paths that define the scope of the virtual stream.
4. Complete the stream settings just as you would any other stream.
5. Click **OK**.

Note

Just as you can create a virtual stream as the child of any stream type, you can also create a real stream as the child of a virtual stream.

For more information about using the Stream dialog to create or edit streams, see [Creating Streams](#).

Submit changes to a virtual stream

You submit changes to a virtual stream just as you would to a real stream, however the changes are actually submitted to the virtual stream's base parent.

Merge and copy to a virtual stream

You merge changes down or copy changes up to a virtual stream just as you would to a real stream, however the merge or copy is actually submitted to the virtual stream's base parent. You might find it more straightforward to copy and merge directly between real streams.

9 | Using P4V for distributed versioning

This chapter provides an introduction to Distributed Versioning via P4V.

Understanding DVCS and setting up the server	135
Init	135
Read this first	136
Directories and files	136
Add files	137
Clone	137
Prepare to fetch and push content between servers	138
Fetch and push	138
Configure security for fetching and pushing	139
Specify what to copy	139
What do Fetch and Push copy?	139
Resubmit	140
Branches	140
Understanding remote mappings	141

Understanding DVCS and setting up the server

Before you start to work with distributed versioning, it is important to understand certain basic concepts — including distributed versioning architecture and how servers relate to one another in this architecture. Please refer to *Using Helix Core for Distributed Versioning* for information on the distributed model.

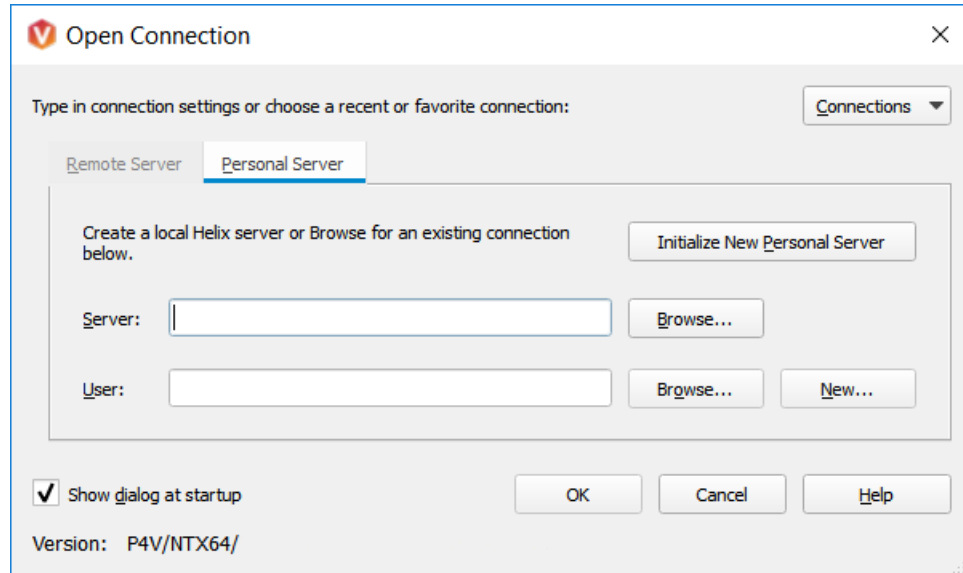
The DVCS functionality of P4V allows users to create personal servers and submit changes locally without the need to connect or set up a remote server. If the user wishes to clone, fetch or push assets from a remote server, that server must meet the minimum version of 2016.2.1487173 and be set up to accept DVCS commands.

Init

This section describes how to start up a personal server using **Init**. Use this approach if you want to work in isolation on a personal server, developing and possibly branching code.

To initialize a personal server and set it up with everything needed to start versioning files, do one of the following:

- On the P4V toolbar, click **Init**.
- Click **Connection > Open Connection**. In the **Open Connection** dialog:
 1. Select the **Personal Server** tab.
 2. Click **Initialize New Personal Server**.



Before you continue, go to "Read this first" below.

For the command-line equivalent of **Init**, see [Initializing a Server](#) in *Using Helix Core for Distributed Versioning*.

Read this first

The Unicode setting and case sensitivity check boxes should match those respective settings on the shared server that will be fetched from and pushed to. If you are unsure what these settings are, the following command-line command will return the pertinent information:

```
p4 -ztag -p <server> info
```

where **<server>** is something like "perforce:1666". P4V 17.1 does not support the auto-discover option for DVCS Init.

Directories and files

Init creates the following directories and files in the directory in which the command is invoked:

- **.p4root** - A directory containing the database files that will contain the metadata about files checked into Helix Server

- **.p4ignore** - A list of files Helix Server should not add or reconcile
- **.p4config** - A file containing configuration parameters for the client-server connection

In addition, **Init** creates:

- A stream depot
- An initial stream called **main**
- A workspace. Note that the client option **allwrite** is set by default, making files writable without the need to check them out first.

Add files

At this point, you are ready to add files to your personal server. You can create, copy, and mark for add your source files to be added to Helix Server and submit them. If you are new to P4V, see "[Managing files](#)" on page 54.

Clone

Clone is a combination of two operations, **Init** and **Fetch**. Clone fetches the files specified in a Remote Mapping and copies them to the new personal server. See "[Understanding remote mappings](#)" on page 141 for more information on Remote Mappings. This approach is best when working collectively on an existing project; users work on a set of project files that are managed on a remote server. They can periodically push changes back to the shared server from which the files were cloned and also periodically fetch to get the latest changes made by others to the shared server files.

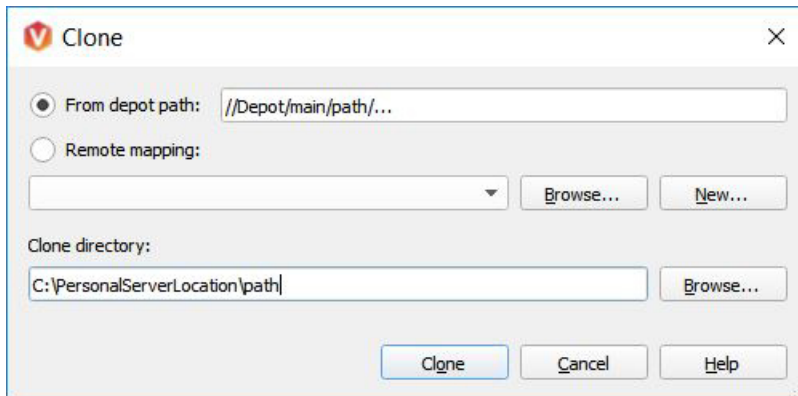
You can perform a Clone operation from a remote server using any of the following methods:

- Connect P4V to a remote Helix server. From the Depot Tree in the left pane, right-click a single path you wish to clone.

With this option, P4V creates a remote mapping on the personal server, called 'origin', for the user that maps to the selected path. There is an option to create or select a different remote mapping.

- Connect P4V to a remote Helix server. On the toolbar, click **Clone**.

With this option, you must specify remote mapping information or create a new remote mapping.



Prepare to fetch and push content between servers

If you subsequently want to push your work to a shared server or fetch files from a shared server, you must create a remote spec. See "Fetch and push" below and "Understanding remote mappings" on page 141 for more information.

Fetch and push

Fetch and **Push** lie at the heart of a collaborative distributed workflow; they enable users to perform a number of major tasks:

- To copy changelists from a personal server to a shared server
- To fetch changelists from a shared server that were pushed there by other personal servers
- To obtain and work with a subset of a shared server's entire repository.
- To copy work between two personal servers

Administrators can also use **Fetch** and **Push** to copy changelists between shared servers.

Fetch and **Push** are to the distributed versioning model what **sync** and **submit** are to the classic Helix Server central server model.

Fetch copies the specified set of files and their history from a remote server into a personal server.

Push copies the specified set of files and their history from a local server to a remote server. Both are atomic: either all the specified files are fetched or pushed or none of them are.

If **Push** fails after it has begun transferring files to the remote server, it will leave those files locked on the remote server. The files cannot be submitted by any other user. If **Push** cannot be quickly retried, you can connect to the remote server with another P4V window to manually unlock the affected files.

When a DVCS repository is made via **Clone** or **Init**, the files in the repository (whether cloned, or fetched from a remote) default to *allwrite* mode. This means the files may be edited without checking them out.

If a file has been changed in *allwrite* mode, P4V treats it like an "offline edit" and will automatically reconcile the change before running **Fetch** or **Push**. This means that the changes are submitted to the repository before the **Fetch** or **Push**.

However, a file which is in a pending changelist (that hasn't been submitted) must be submitted before running **Fetch** or **Push**. If files are shelved, they should be unshelved and submitted, or deleted before running **Fetch** or **Push**.

Configure security for fetching and pushing

In order to **Fetch** and **Push** between servers, the respective servers must have authentication and access permissions configured correctly:

- The user name on the remote server must be the same as the user name on the local server. This will be the case by default unless you have specified the **Remote User** field in the remote server's remote mapping.
- The user must exist on the remote server.
- The user must have *read* (**Fetch**) and *write* (**Push**) permission on the remote server.
- The **server.allowpush** and **server.allowfetch** configuration settings must be set to On (they are Off by default) on both the remote server and the personal server.

Specify what to copy

If the local and remote sides of the Depot Map pattern is modified to map differently within the Remote Map, and a filespec or stream name is provided, then the filespec argument or stream name must be specified using the personal server's depot syntax. The filespec must always use depot syntax, not file system or client syntax. For more information, see "[Understanding remote mappings](#)" on page 141.

What do Fetch and Push copy?

In addition to the specified set of files, the changelists that submitted those files, and integration records, fetching and pushing to a server also copies the following:

- attributes
- any fixes associated with the changelists, but only if the job that is linked by the fix is already present in the local server

Note

Zipping and unzipping files also copies attributes and fix records.

Resubmit

If there are file conflicts between the personal and remote servers when performing a fetch those conflicts will enter a resolve/resubmit cycle. In the cycle, one or more conflicting files from the remote and personal server are resolved, then the resolved files are resubmitted.

There is one resubmit for each affected changelist that is affected by the fetch.

The resolve/resubmit cycle is automated and can be completed without user interaction but if a file conflict exists each cycle will require user interaction.

The Resubmit dialog will appear if the fetch has conflicts. At the top is a list of all of the affected changelists along with their resubmit status. Below that list is the current changelist that is being resolved.

The user can interrupt the resolve/resubmit cycle at any time by choosing "Finish Later". It is highly recommended that a user complete the cycle before performing any actions to affected files to minimize potential data loss. P4V will reopen the resubmit menu if it has been restarted.

If a file needs to be resolved before resubmitted, the Resubmit dialog will provide an option to resolve the file conflict. The P4V Resolve dialog will load prepopulated with the files associated in the current step of the resolve/resubmit cycle.

The sense of the resolve for a DVCS Fetch with conflicts is the reverse of what might be expected. When the Fetch brings down files from the remote server, all local files in conflict are moved to a "tangent" depot residing on the local server. Moving files to the tangent depot is automatically done as part of the Fetch process. For each changelist with conflicts, a new changelist containing changes from the remote server replaces the old changelist that has been moved to the tangent depot. The "source" for the resolve refers to the local files in the tangent depot. The "target" for the resolve refers to the remote files that have been moved to the local depot itself. For example, in P4Merge, the "source" is the local file (say MyFile.c#7); the "target" is the remote file with conflicts (with no stated revision); and the "base" is the common previous revision (say MyFile.c#6). If there's any confusion, running P4Merge from the Resolve dialog (by selecting the "Merge Tool" option) should clarify the sense of the "source" and "target" of the resolve.

Branches

Branches within a Helix Server personal server are implemented as a stream. For more information on Streams and Branching, please review "[Working with streams](#)" on page 107. The branch operation in Streams is the same as the **p4 switch** operation. In P4V, branching will create a new Stream of type main or development.

Users familiar with Helix Server streams: When connected to a personal server, the P4V Stream Graph View will change streams-specific terminology to include git-related terminology. Example: a stream is called a "Branch Stream". Switching streams is renamed to "Checkout Stream".

To view the branches of your personal server, open the Stream Graph View by selecting Streams View (Ctrl+7) from the View menu. Under Graph View Options, make sure your stream(s) are selected, and click Apply.

For more information on the Stream Graph View including information on the display conventions and configuration, please see ["Using the stream graph" on page 119](#).

To branch in P4V, from the Stream Graph View, context-click on the stream you wish to branch and select Branch Stream.

When creating a new branch, files that have been modified in the current branch are shelved for safekeeping. After switching back to an existing branch, P4V syncs your client workspace to the head of the new branch and unshelves any files that were open in the default changelist the last time the stream was used.

To change your workspace to a different branch either:

- Double-click the branch and select Work in this stream
- Right-click the branch and select Checkout Stream
- Drag the workspace icon from the stream you are currently working in to another stream

Note You cannot switch to a new branch if files are open in a numbered changelist. If files are open in the default changelist, they will be shelved and reverted prior to switching to the new branch, and will be automatically unshelved when switching back to thi

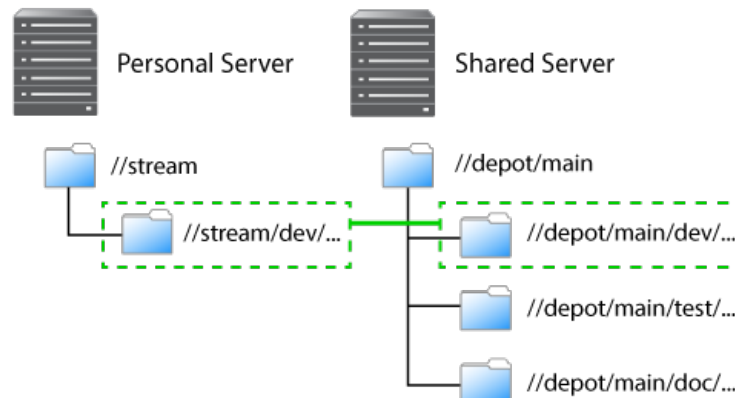
Note

You cannot switch to a new branch if files are open in a numbered changelist. If files are open in the default changelist, they will be shelved and reverted, prior to switching to the new branch, and will be automatically unshelved when switching back to this branch.

Understanding remote mappings

A remote describes how depot files are mapped between a personal server and a shared server. A remote spec — which describes a remote — is created by the user and has a unique name. A remote is used with **Push**, **Fetch**, and **Clone** to describe source and target directories.

The following picture illustrates mapping depot files between a personal and a shared server:



As depicted in the figure above, a remote holds file mappings between depot paths on the shared server and depot paths on the personal server.

- For fetch and clone operations, it defines the files from the remote server that you want in your personal server and specifies where you want them to reside.
- For a push operation, it defines the files from the personal server that you want in the shared server and specifies where you want them to reside.

Remotes provide a convenient way to give you the exact files you need to work on a particular project. You can simply clone from a shared server, specifying the remote id of the remote that maps the desired files. These files are then copied to your personal server. Once they've cloned, you can **Fetch** to refresh the files initially obtained with **Clone**.

Using remotes allows you to fetch a subset of all the files on the shared server. This is in contrast to other distributed versioning systems, such as Git, which require that you fetch all files.

Note that when you clone a set of files from a shared server by specifying a remote, Helix Server creates a new remote named *origin* and copies the remote into your local system. This remote will be used for all future fetches.

There are two different scenarios in which remotes are created:

- You create a remote on a shared server so that other users can clone from this server and obtain the files they need to work on a project. Note that to create a remote on a shared server, you must have an access privilege of **open** or greater. While this task typically falls in the domain of an administrator, it does not require administrator privileges.
- You — an individual user — create one or more remotes on your personal server so that you can eventually push your work to and fetch files from one or more shared servers.

You would create a remote on a shared server to dictate which subset of the shared server's repository a personal server retrieves when it clones from the shared server. After cloning, you use the *origin* remote on your personal server. You can then either edit the *origin* remote or create a different remote to control which streams the personal server fetches and pushes.

10 | Integration with Swarm

Helix Swarm is a powerful and flexible code review and collaboration solution that helps teams ship quality software faster. Swarm enables review of code and other assets before or after commit and can be customised to fit into various workflows. Swarm stores all of its metadata (including Reviews, Projects and Comments) in Helix server, which makes it an attractive solution because it does not require backing up of an external database. For more about using and installing Swarm, see the [Helix Swarm Guide](#).

Once the Swarm integration is enabled, a number of additional features are available in P4V, including new context menus, request review and update review dialogs, badging on pending and committed changes, as well as **Review ID** and **Review State** columns.

Review workflow

Following is the happy path workflow for a Swarm review. For more permutations and variations, see the [Helix Swarm Guide](#).

1. **Make local changes to files:** Swarm reviews can follow either a pre-commit or post-commit workflow. In both models, the author would make some local content changes to one or more files and then get those content changes into Perforce.
2. **Request a review:** For pre-commit code reviews, the Swarm solution uses Helix Server shelving technology to get the content to Helix Server. For post-commit code reviews, content committed to Helix Server is added to a review. In both cases, a Swarm review is created with an ID, a description, a set of files, and other meta-data including the author, reviewers, and comments made on the review.
3. **Provide review feedback:** Reviewers can comment on files or individual lines in files using Swarm. They can also add follow-up tasks that the author should address before the review can be closed.
4. **Request revisions:** If the reviewers find that the review needs more work, which is often the case, they can change the state of a review to **Needs Revision**, thereby notifying the author that the review is back in their court.
5. **Request further review:** Authors can request another round of review of their review content changes and update any of the tasks they were asked to complete, thereby notifying the reviewers that they are ready for more feedback.
6. **Approve or reject review:** Reviewers can approve or reject a review using Swarm. Once they approve or reject a review, the review is considered closed.
7. **Commit the review:** For pre-commit reviews, authors can commit reviews either using Swarm or their Helix Server clients, such as P4V or P4VS. For the latter scenario, committing a pre-commit code review is synonymous with submitting the changelist associated with the review.

Setting up the Swarm integration

To use the full list of features of the P4V integration with Swarm, you need to have Swarm 2014.4 or later installed. The P4V integration also works with Swarm 2014.3, but some features, such as the required reviewers option, are not visible. For details on how to set up the Swarm integration, see [Configuring Swarm connections](#) in the *Helix Versioning Engine Administrator Guide: Fundamentals*.

Example **p4 property** command to run:

```
p4 property -a -n P4.Swarm.URL -v https://swarm.yourcompanydomain.com
```

where *https://swarm.yourcompanydomain.com* is the URL for your Swarm server.

If you are testing the Swarm integration, you may wish to set the property for a specific user. For example, to enable the Swarm integration for the user **username**:

```
p4 property -a -u username -n P4.Swarm.URL -v  
https://swarm.yourcompanydomain.com
```

Similarly, you can enable the Swarm integration for a specific group of users. For example, to enable the Swarm integration for the group **group**:

```
p4 property -a -g group -n P4.Swarm.URL -v  
https://swarm.yourcompanydomain.com
```

Note

P4V uses a Swarm integration timeout, in seconds, to limit delays in the P4V user interface. The timeout can be adjusted by running the following command:

```
p4 property -a -n P4.Swarm.Timeout -v 10
```

The default timeout value is 10 seconds. The timeout can be set for a specific user or a specific group by including the **-u username** or **-g group** options, respectively.

Swarm integration features

Once the Swarm integration is enabled, a number of additional features are available in P4V, including new context menus, request review and update review dialogs, badging on pending and committed changes, as well as **Review ID** and **Review State** columns.

Request a review

You can request reviews from either pending or submitted changelists. Note that a changelist cannot be associated with more than one review. However, a review can have more than one changelist associated with it.

Best practices:

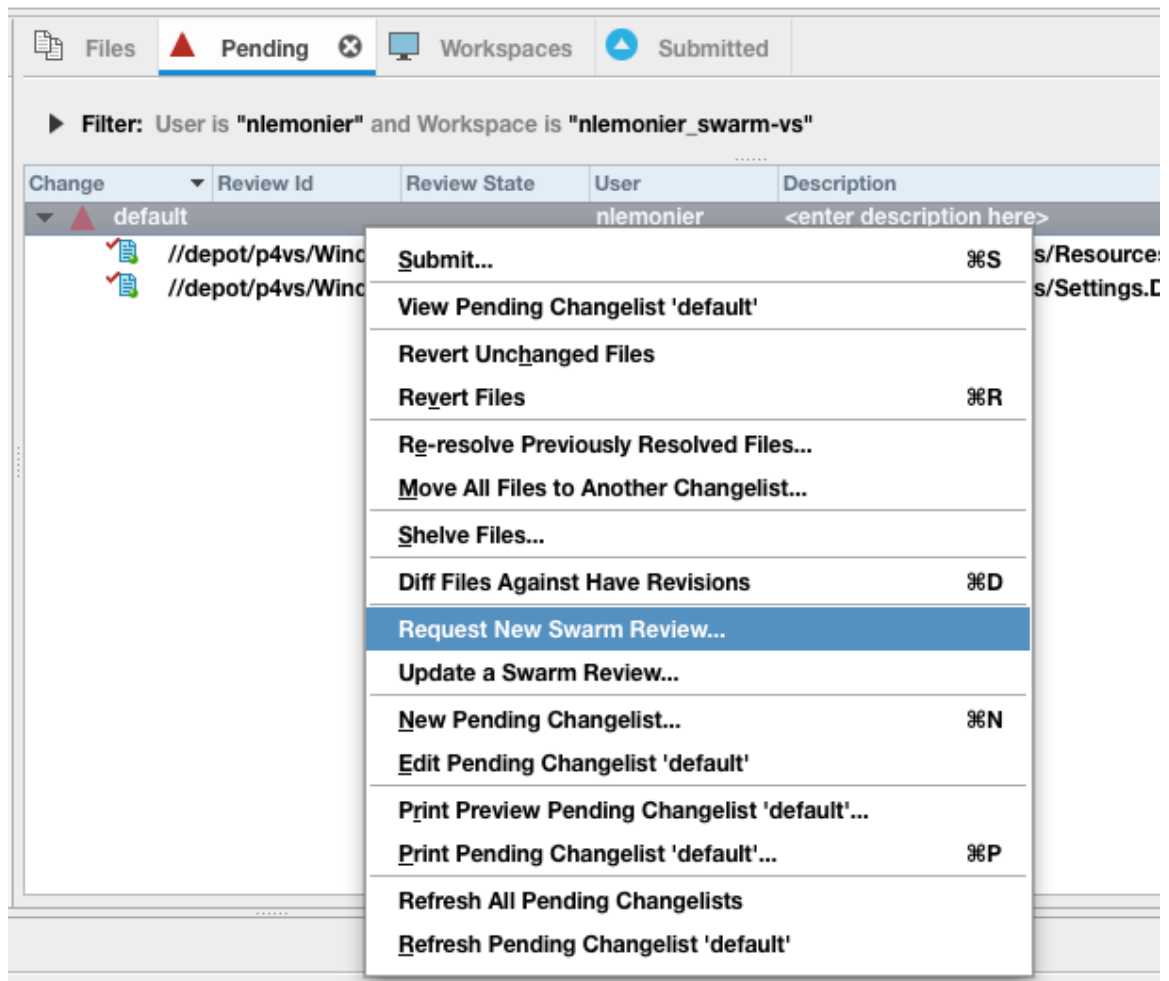
- Pre-commit code reviews are a more popular approach because they allow validating of code and correcting defects before they become a part of the committed code-base. Swarm supports pre-commit code reviews via pending changelists.
- Post-commit code reviews allow reviewers to provide feedback on the submitted content, and they allow the author to follow on with more submitted changes when making updates recommended by the reviewers. Development branches are well-suited for the post-commit review process.

Request a review from a pending changelist

To request a review from a pending changelist, do the following:

1. Go to the pending changelist tab, right-click the changelist, and select **Request New Swarm Review**.

Note that if the changelist is already part of a Swarm review, this option is not available.

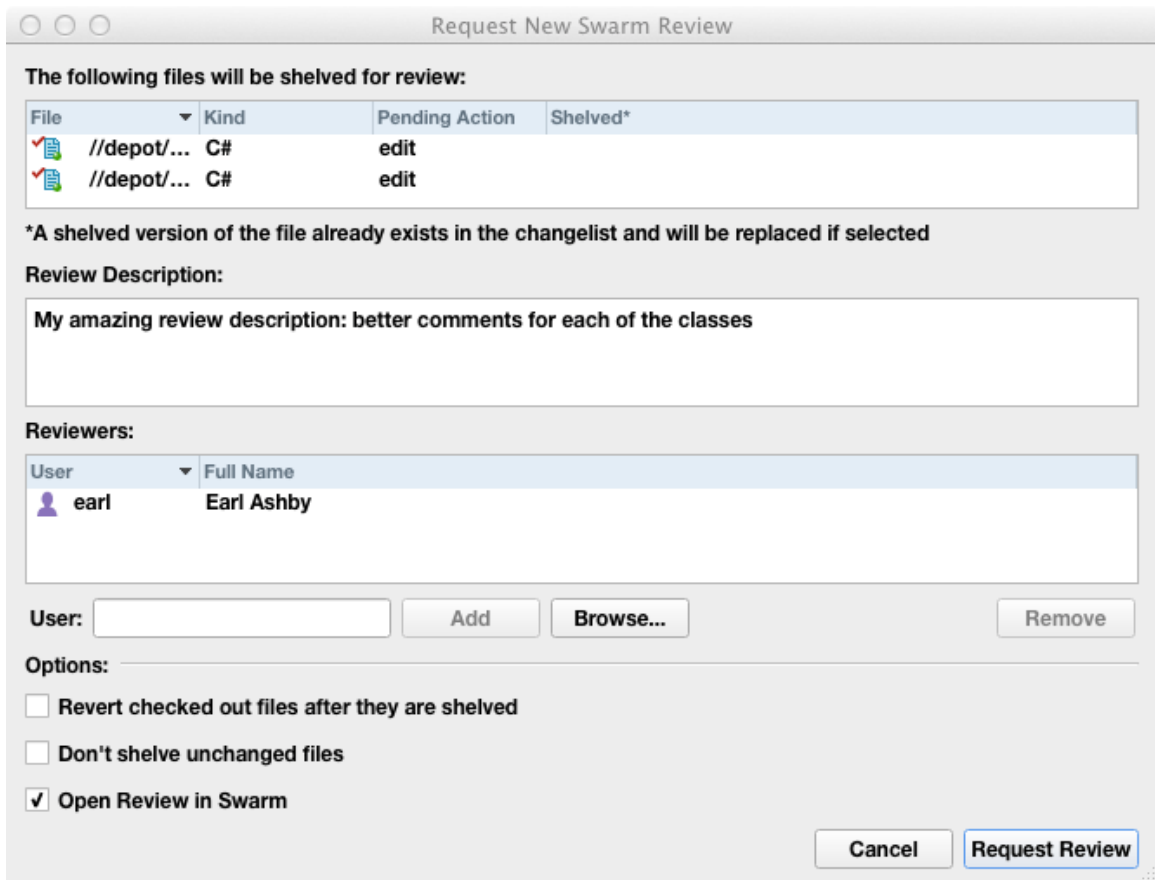


The **Request New Swarm Review** dialog displays a list of files to be shelved in order to request the review. If the changelist already has shelved files, the dialog also lists these already shelved files. The aggregate of the shelved files comprises the review.

2. In the **Request New Swarm Review** dialog, enter a description.



Each review requires a description. The default description is the changelist's description.

3. Optionally, select additional options, including: reviewers, reverting checked out files after they are shelved, not shelving unchanged files, and opening the review in Swarm.



Request New Swarm Review

The following files will be shelved for review:


File	Kind	Pending Action	Shelved*
 //depot/...	C#	edit	
 //depot/...	C#	edit	

*A shelved version of the file already exists in the changelist and will be replaced if selected

Review Description:

My amazing review description: better comments for each of the classes

Reviewers:

User	Full Name
 earl	Earl Ashby

User:

Options:

Revert checked out files after they are shelved

Don't shelve unchanged files

Open Review in Swarm

4. Click **Request Review**.

The pending changelist is badged with a Swarm icon, and P4V updates the **Review Id** and **Review State** fields with their values from Swarm.

Change	Review Id	Review State	User	Description
default			nlemonier	<enter description here>
12822	12823	Needs Review	nlemonier	My amazing review description: better comments for each of ...

Tip

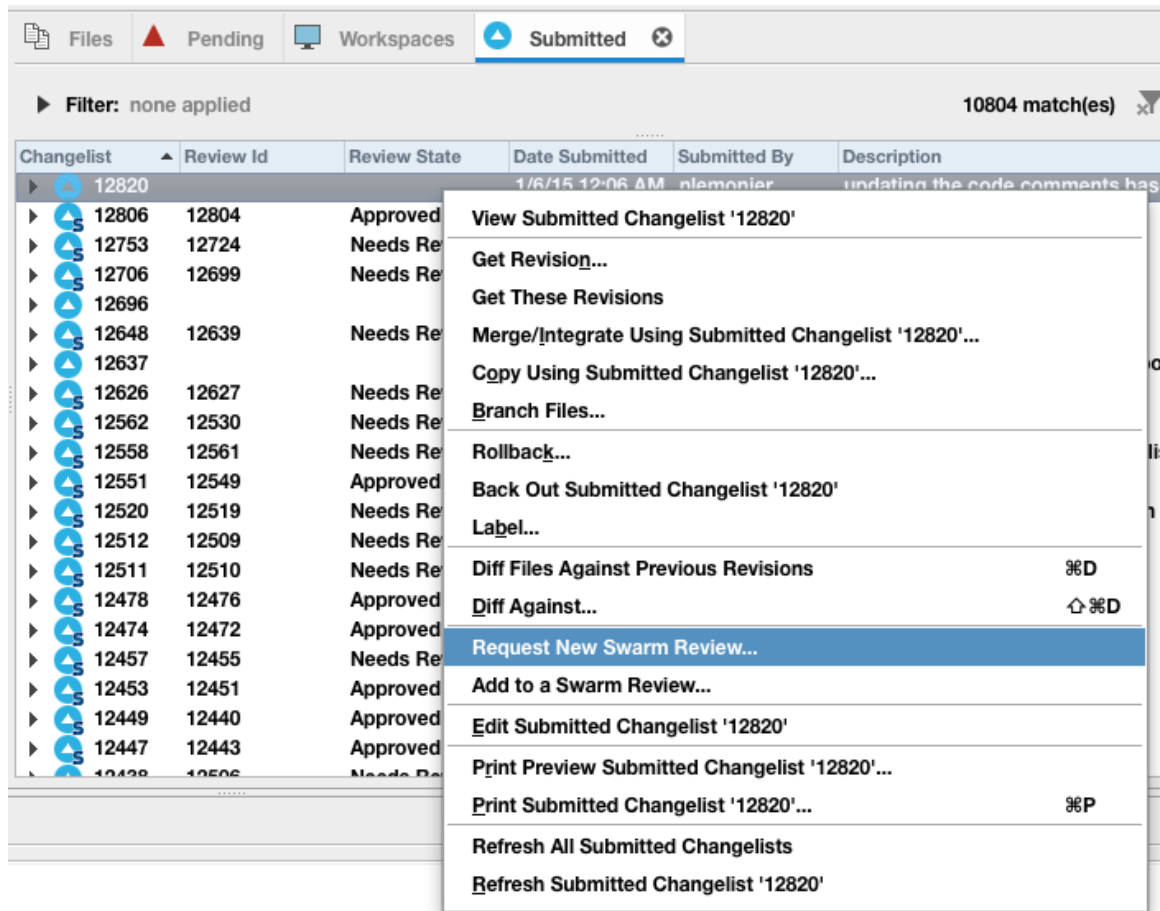
It is a best practice that you, as the author, keep this pending changelist for subsequent updates to the review. You can use this same changelist to submit the review. If the review is rejected or the review is committed from Swarm, you should manually discard this pending change so that it does not get accidentally committed.

Request a review from a submitted changelist

To request a review from a submitted changelist, to the following:

1. Go to the **Submitted changelist** tab, right-click the changelist, and select **Request New Swarm Review**.

Note that if the changelist is already part of a Swarm review, this option is not available.



The **Request New Swarm Review** dialog displays the files to be added to the review.

2. In the **Request New Swarm Review** dialog, enter a description.

Each review requires a description. The default description is the changelist's description.

- Optionally, select additional options, including reviewers and opening the review in Swarm.

Request New Swarm Review



The following files will be sent for review:

File Name	Revision	Action	FileType	In Folder
<input checked="" type="checkbox"/> AssemblyInfo.cs	3	edit	text	//depot/p4vs/WindowsFormsA...
<input checked="" type="checkbox"/> Resources.Desi...	3	edit	text	//depot/p4vs/WindowsFormsA...

Review Description:

updating the code comments based on feedback

Reviewers:



User	Full Name
 bruno	bruno
 earl	Earl Ashby

User:

Open Review in Swarm

- Click **Request Review**.

The pending changelist is badged with a Swarm icon, and P4V updates the **Review Id** and **Review State** fields with their values from Swarm.

Files	Pending	Workspaces	Submitted	
Filter: none applied				
Changelist	Review Id	Review State	Date Submitted	Submitted By
 12820	12821	Needs Review	1/6/15 12:06 AM	nlemonier
 12806	12804	Approved	1/2/15 1:55 PM	crank

Update a review

If you need to update the files in a review for any reason, such as to respond to the feedback you received from the reviewers, P4V lets you update an existing Swarm review.

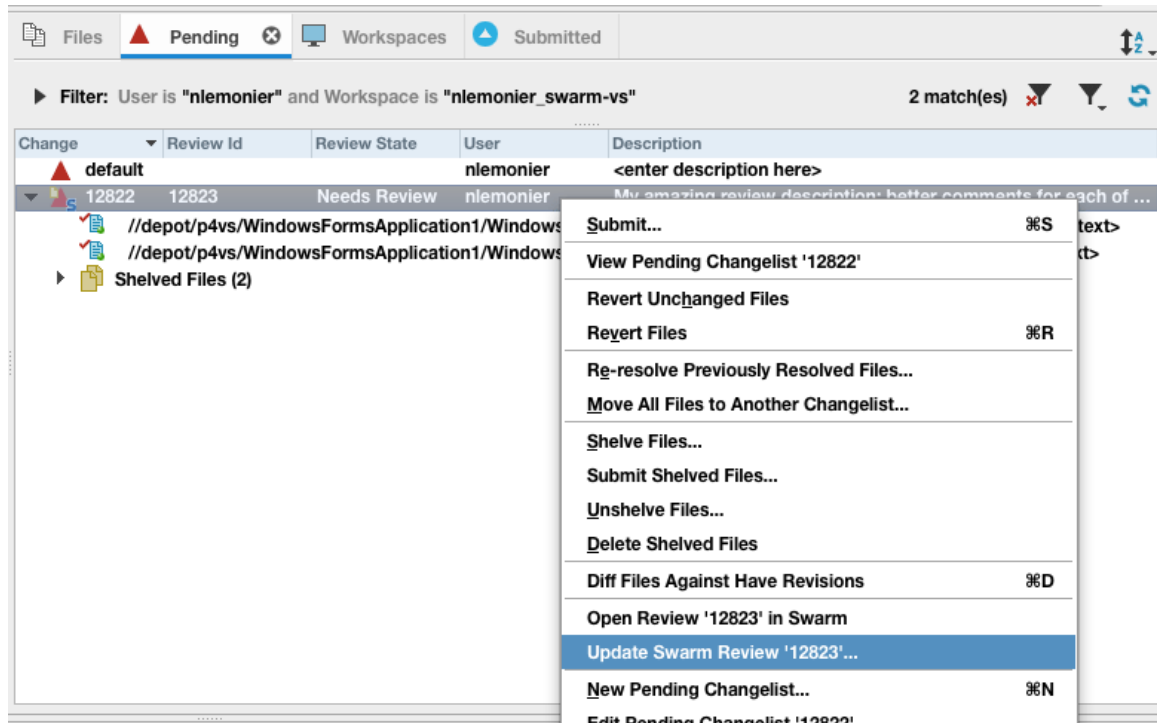
Update a Swarm review from a pending changelist

When updating a Swarm review from a pending changelist, the changelist can – but does not have to be – already associated with the review.

Option 1: Update from an associated pending changelist

To update a review from a pending changelist that is associated with the review, do the following:

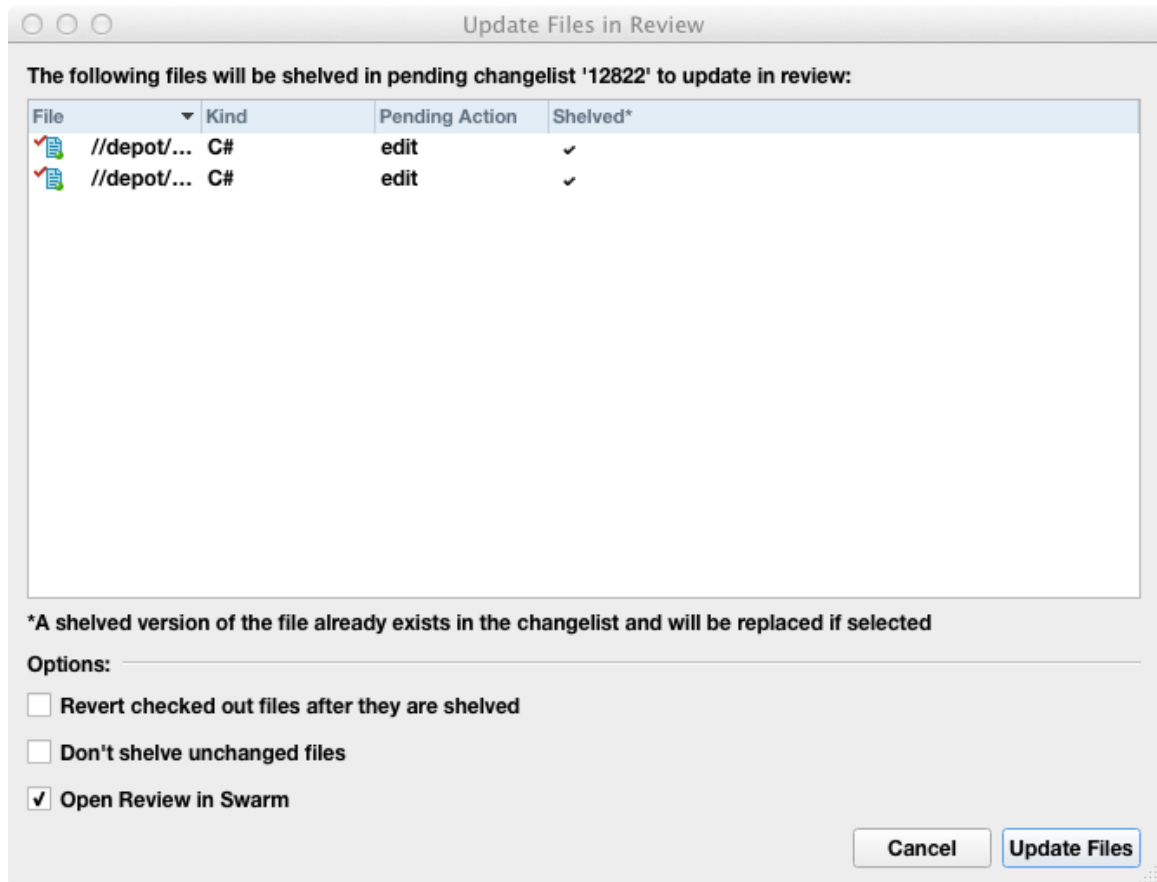
1. On the **Pending** tab, right-click the changelist and select **Update Swarm Review 'xxxx'**, where xxxx is the review ID.



The **Update Files in Review** dialog displays a list of files to be shelved in order to update the review. If the changelist already has shelved files, the dialog also lists these already shelved files. The aggregate of the shelved files comprises the updated review.

2. If needed, update the review description.

- Optionally, select additional options, including reverting checked out files after they are shelved, not shelving unchanged files, and opening the review in Swarm.



- Click **Update Files**.

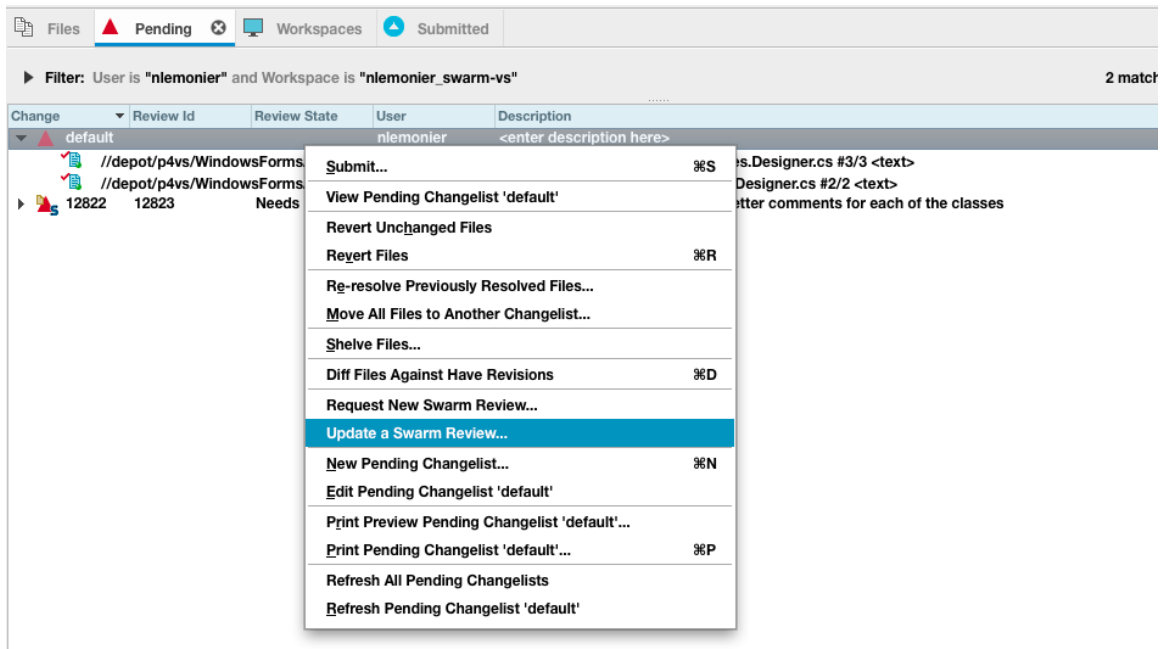
The files in the associated review are updated.

Option 2: Update from a pending changelist not associated with the review

To update a review from a pending changelist that is not yet associated with the review, do the following:

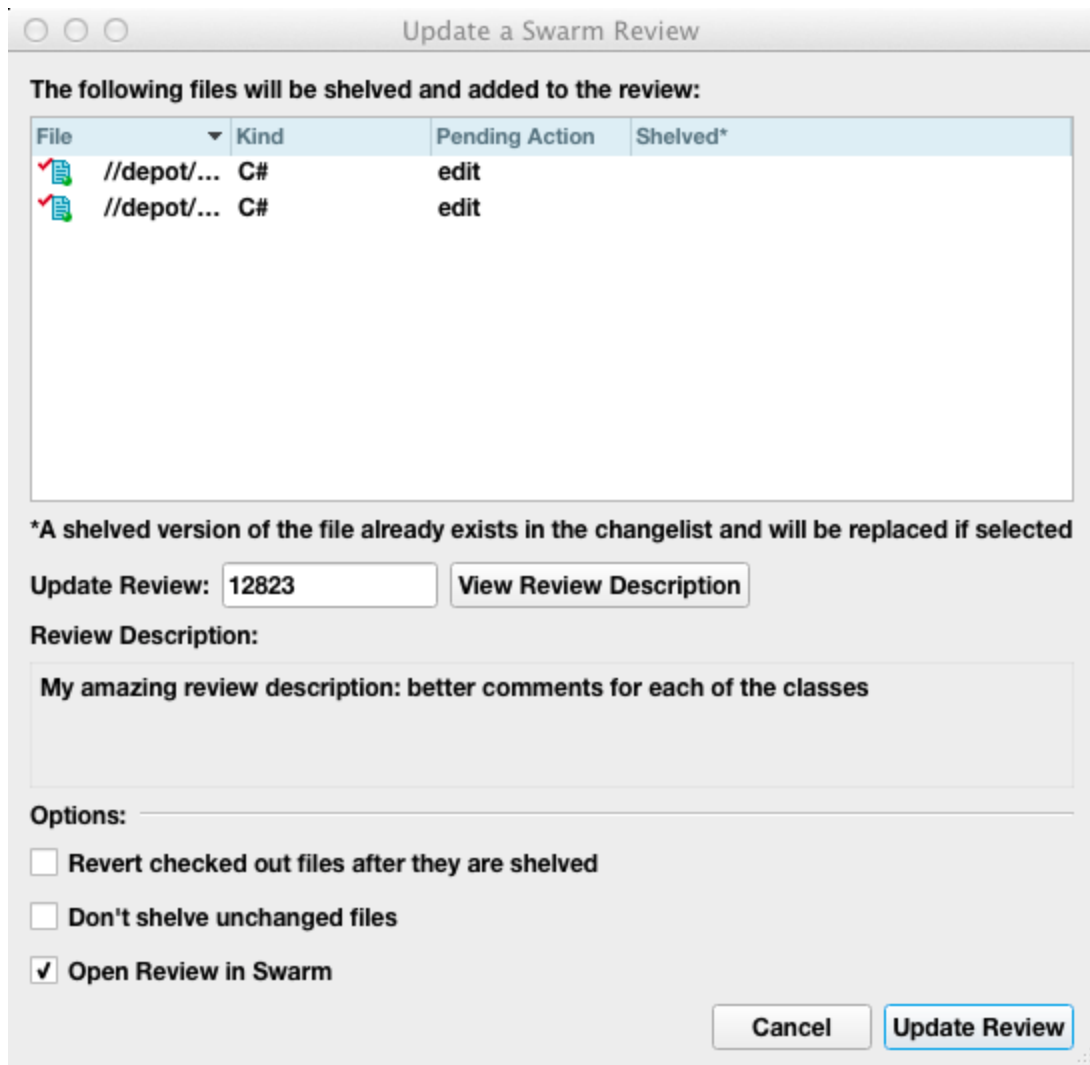
- On the **Pending** tab, right-click the changelist and select **Update Swarm Review**.

This scenario is typical if you are working on a different machine than where you originally requested the review or if you have discarded the original changelist you used for creating the review. If you already have a changelist associated with the review, you would likely use that changelist and follow the instructions in [option 1](#).



The **Update Files in Review** dialog displays a list of files to be shelved in order to update the review.

- In the **Update Review** field, enter the ID of the review you would like to update with these files.



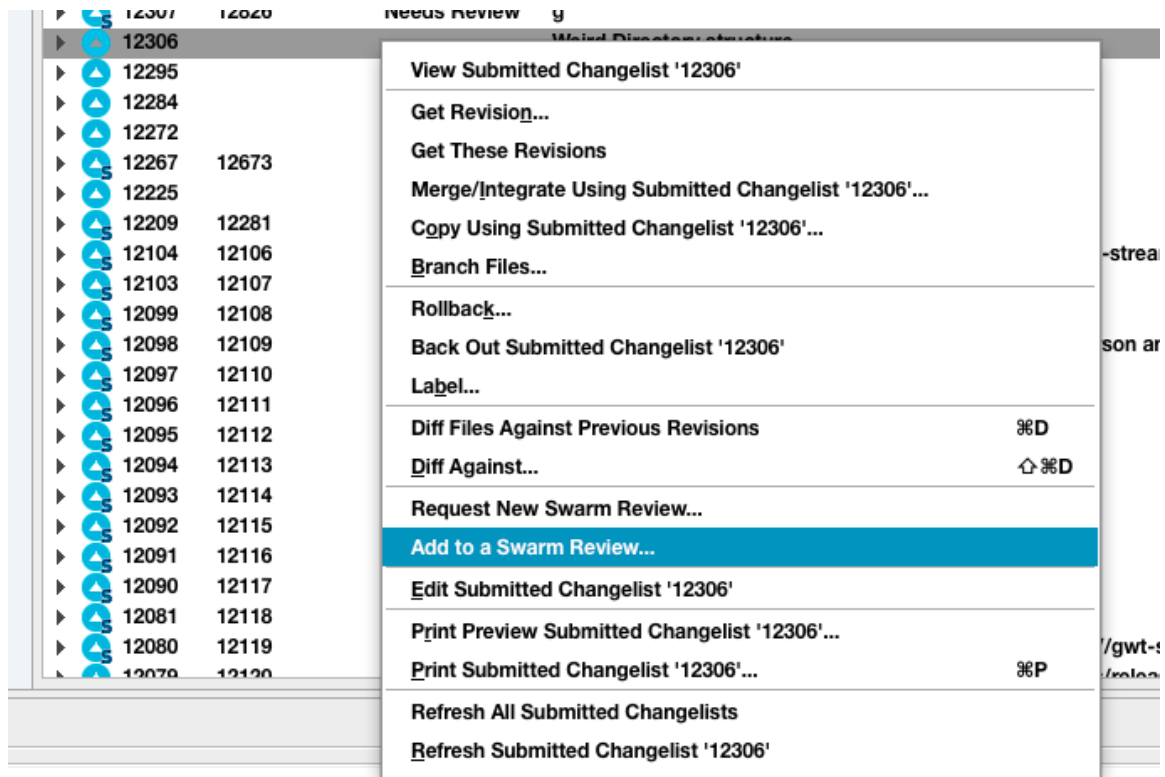
- Optionally, to validate if this review is in fact the review you would like to update, click **View Review Description**.
- Optionally, select additional options, including reverting checked out files after they are shelved, not shelving unchanged files, and opening the review in Swarm.
- Click **Update Review**.

The changelist you used to update the review becomes associated with the review.

Update a Swarm review from a submitted changelist

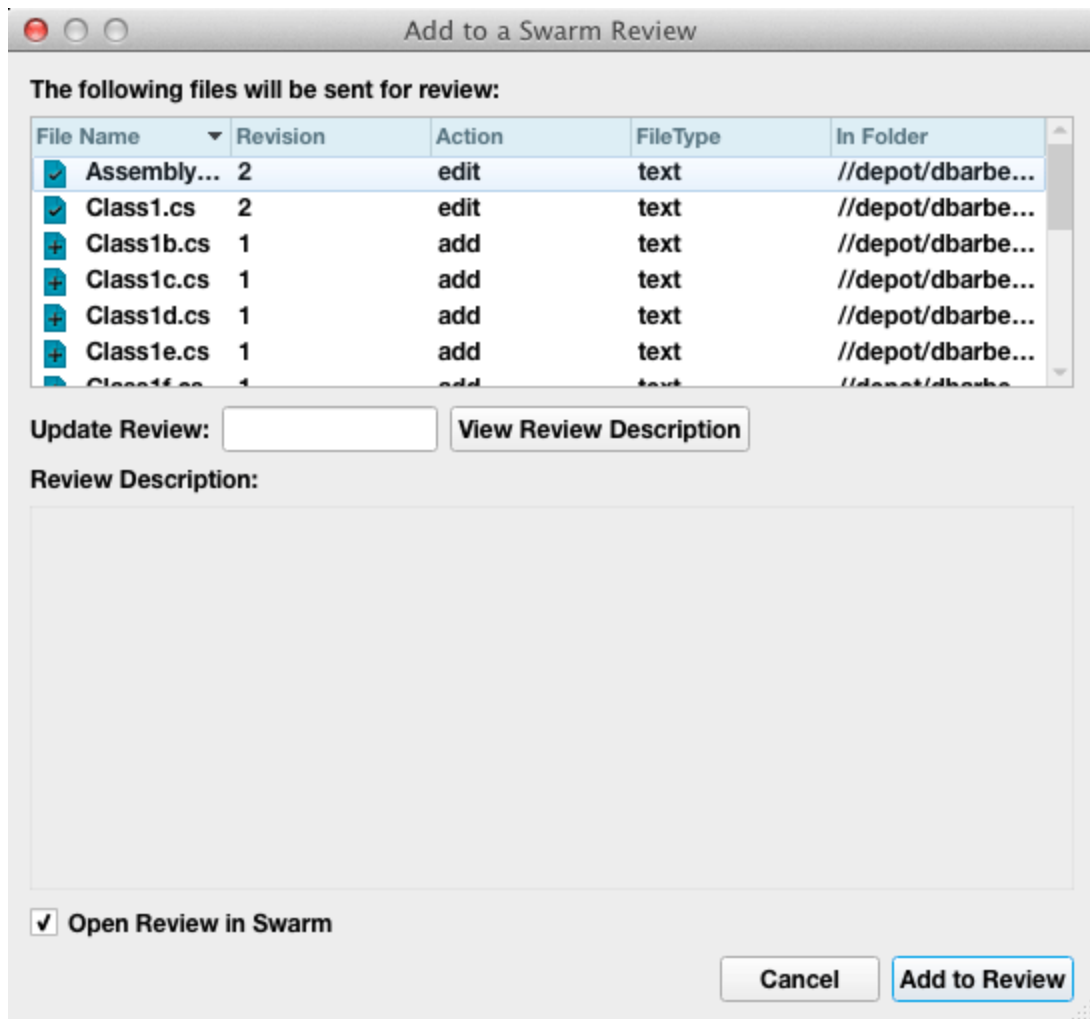
To associate a submitted changelist with an existing Swarm review, do the following:

1. Right-click the submitted changelist and select **Add to Swarm Review**.



The **Add to a Swarm Review** dialog displays a list of files to be added to the review.

- In the **Update Review** field, enter the ID of the review to which you would like to add these files.



- Optionally, to validate if this review is in fact the review to which you would like to add these files, click **View Review Description**.
- Optionally, select additional options.
- Click **Add to Review**.

The files get added to the review.

Open a review in Swarm

If you leave the **Open Review in Swarm** check box option selected in the **Review Request** or **Review Update** dialogs, P4V launches Swarm to the review page in your default browser. This serves as confirmation that the review has been created or updated.

Review 12823

nlemonier authored this 15 minutes ago for //depot/p4vs/WindowsFormsApplication1/WindowsFormsApplication1/Properties, last updated 15 minutes ago

Needs Review

My amazing review description: better comments for each of the classes

+ Add Job

Tasks: 0 (red), 0 (green), 0 (blue)

Reviewers: 0 (up), 0 (down)

Files: 2, Comments: 0, History

#: Change 12822 shelved into //depot/p4vs/WindowsFormsApplication1/WindowsFormsApplication1/Properties (2 added, 0 removed, 0 changed)

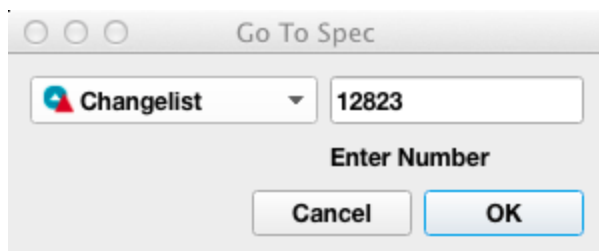
- Resources.Designer.cs
- Settings.Designer.cs

Tip: Use [n] and [p] to cycle through the changes.

Getting Files from a Review

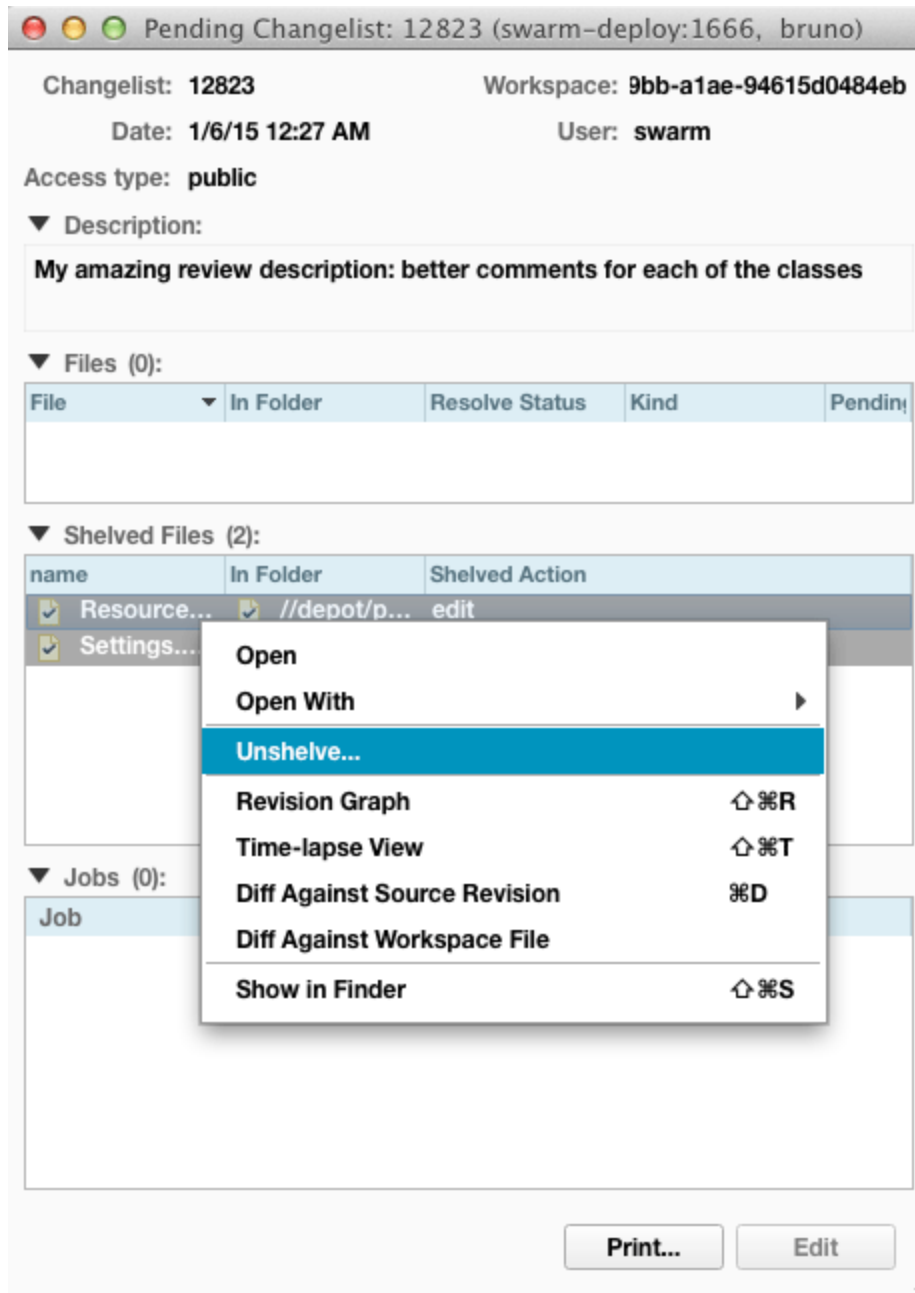
As a reviewer, you may be inspecting a review in Swarm and decide you need to get a local copy of the files. The review ID shown in Swarm corresponds to a pending changelist in Helix Server that contains these files. The simplest way to get these files from P4V is to use the **Go To Spec** dialog, as follows:

1. In P4V, open the **Go To Spec** dialog with the keyboard shortcut of **+G** (+G on Mac).

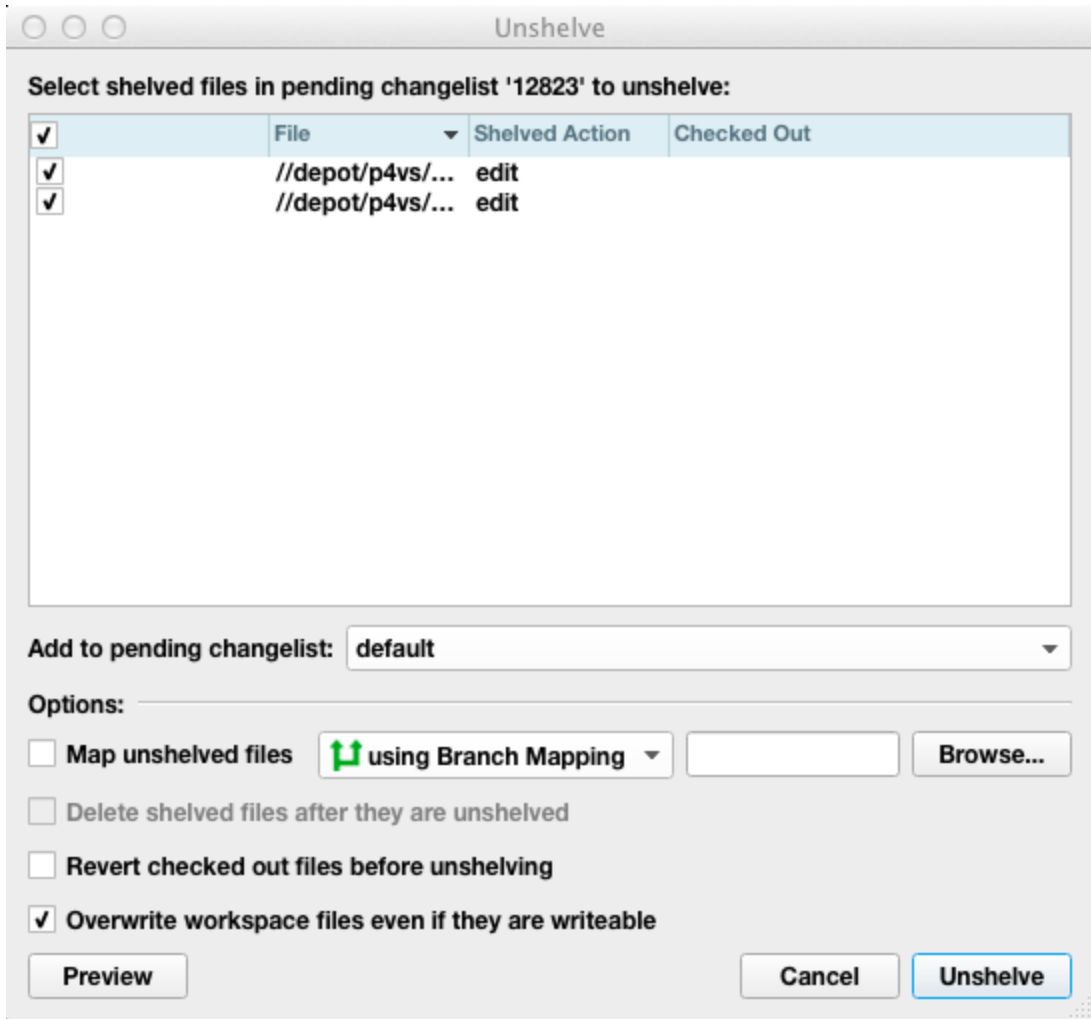


2. Enter the review ID and click **OK**.
P4V displays the **Pending Changelist** dialog.

3. Right-click the files you want to unshelve and select **Unshelve**.



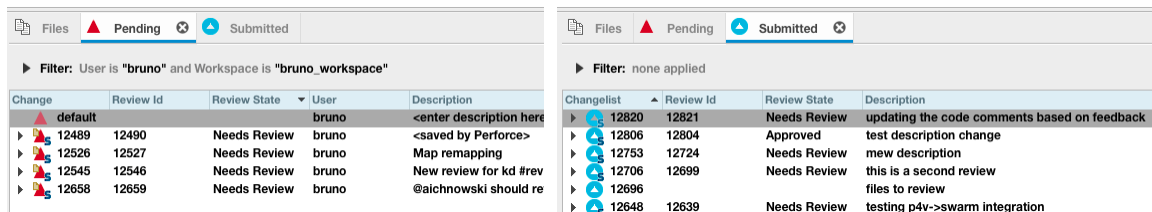
P4V displays the **Unshelve** dialog.



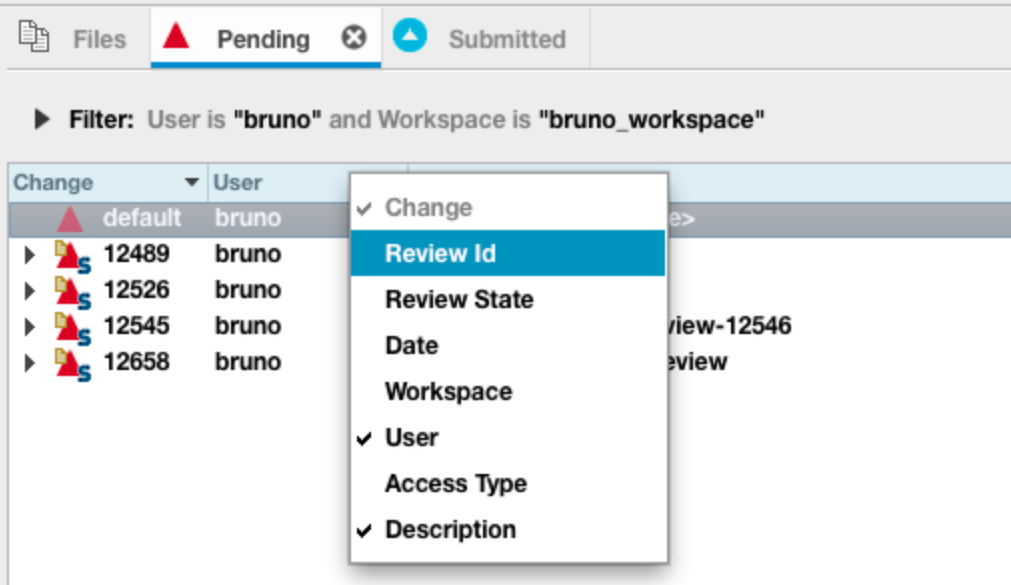
4. Select the pending changelist to which you want these files to be unshelved and click **Unshelve**. P4V retrieves a local copy of these files into your workspace.

Review ID and Review State columns

P4V adds a **Review Id** and **Review State** column to both the submitted and pending changelist tabs for connections that have the Swarm integration enabled.



If you are connected to a Helix server with the Swarm integration enabled and do not see the columns, right-click the header row and select these fields.



Glossary

A

access level

A permission assigned to a user to control which commands the user can execute. See protections.

admin access

An access level that gives the user permission to run commands that override metadata but do not affect the state of the service.

APC

APC is the Alternative PHP Cache, a free, open, and robust framework for caching and optimizing PHP intermediate code.

apple file type

Helix server file type assigned to files that are stored using AppleSingle format, permitting the data fork and resource fork to be stored as a single file.

atomic change transaction

Grouping operations affecting a number of files in a single transaction. If all operations in the transaction succeed, all the files are updated. If any operation in the transaction fails, none of the files are updated.

B

base

The file revision on which two newer, conflicting file revisions are based.

BCC

Blind carbon copy, a feature of email, allows the sender of a message to conceal the person entered in the Bcc: field from all other recipients.

binary file type

Helix server file type assigned to a non-text file. By default, the contents of each revision are stored in full, and the file is stored in compressed format.

BMP

Bitmap, a graphics file format, commonly with the filename extension .bmp.

branch

(noun) A set of related files created by copying files, as opposed to adding files. A group of related files is often referred to as a codeline. (verb) To create a stream by copying another stream with p4 copy or p4 populate.

Branch

A branch in Perforce is a set of related files that exist at a specific location in the Perforce depot as a result of being copied to that location, as opposed to being added at that location.

branch form

The form you use to modify a branch.

branch mapping

Specifies how a branch is to be created by defining the location of the original codeline and the branch. The branch mapping is used by the integration process to create and update branches. Client workspaces, labels, and branch specifications cannot share the same name.

branch view

A specification of the branching relationship between two codelines in the depot. Each branch view has a unique name and defines how files are mapped from the originating codeline to the target codeline. See branch.

C

change review

The process of sending email to users who have registered their interest in changes made to specified files in the depot.

changelist

An atomic change transaction in Helix server. The changes specified in the changelist are not stored in the depot until the changelist is submitted to the depot.

Changelist

A changelist is a list of files, their version numbers, the changes made to the files, and a description of the changes made. A changelist is the basic unit of versioned work in a Helix Versioning Engine.

changelist form

The form you use to modify a changelist.

changelist number

The unique numeric identifier of a changelist.

checkpoint

A copy of the underlying metadata at a particular moment in time. See metadata.

classic depot

The depot name that is assumed when no name is specified. The default depot name is depot. See also default depot and stream depot.

client form

The form you use to define a client workspace.

client name

A name that uniquely identifies the current client workspace.

client root

The root directory of a client workspace. If two or more client workspaces are located on one machine, they cannot share a root directory.

client side

The right-hand side of a mapping within a client view, specifying where the corresponding depot files are located in the client workspace.

client workspace

Directories on your machine where you work on file revisions that are managed by Helix server. By default, this name is set to the name of the machine on which your client workspace is located; to override the default name, set the P4CLIENT environment variable. Client workspaces, labels, and branch specifications cannot share the same name.

Code Review

A code review in Swarm is a process in which other developers can see your code, provide feedback, and approve or reject your changes.

codeline

A set of files that evolve collectively. One codeline can be branched from another, allowing each set of files to evolve separately.

Comment

A comment in Swarm is feedback provided on a changelist or a file within a change.

conflict

One type of conflict occurs when two users open a file for edit. One user submits the file, after which the other user cannot submit because of a conflict. The cause of this type of conflict is two users opening the same file. The other type of conflict is when users try to merge one file into another. This type of conflict occurs when the comparison of two files to a common base yields different results, indicating that the files have been changed in different ways. In this case, the merge can't be done automatically and must be done by hand. The type of conflict is caused by non-matching diffs. See file conflict.

counter

A numeric variable used by to track changelist numbers in conjunction with the review feature.

CSRF

Cross-Site Request Forgery, a form of web-based attack that exploits the trust that a site has in a user's web browser.

D

Debian

Debian GNU/Linux is a distribution of the Linux operation system produced by the Debian Project. Debian is also the casual name of the software packaging system used by the Debian GNU/Linux, and its variants such as Ubuntu Linux.

default changelist

The changelist used by commands, unless a numbered changelist is specified. A default pending changelist is created automatically when a file is opened for edit.

default depot

The depot name that is assumed when no name is specified. The default depot name is depot.

deleted file

In Helix server, a file with its head revision marked as deleted. Older revisions of the file are still available.

delta

The differences between two files.

depot

A file repository hosted on the server. It contains all versions of all files ever submitted to the depot. There can be multiple depots on a single installation.

Depot

A depot is a top-level unit of storage for versioned files within a Perforce server.

depot root

The root directory for a depot.

depot side

The left side of any client view mapping, specifying the location of files in a depot.

depot syntax

Helix server syntax for specifying the location of files in the depot.

detached

A machine that cannot connect to the server.

diff

(noun) A set of lines that don't match when two files are compared. A conflict is a pair of unequal diffs between each of two files and a common third file. (verb) To compare the contents of files or file revisions.

donor file

The file from which changes are taken when propagating changes from one file to another.

E

Emoji

Emoji is a Japanese term for ideograms, emoticons, or smileys that can be used to express emotions in electronic messages. See the [Emoji Cheat Sheet](#) for Emoji common to many web applications.

Emoticon

A pictorial representation an author's facial expression (or other symbology) indicating the mood or temper of a portion of text. These include *smileys*, such as :-), to indicate happiness, humor, disapproval, etc. For more information, see [Wikipedia](#).

EPS

Encapsulated PostScript, a graphics file format, commonly with the filename extension .eps.

exclusionary access

A permission that denies access to the specified files.

exclusionary mapping

A view mapping that excludes specific files.

F

file conflict

In a three-way file merge, a situation in which two revisions of a file differ from each other and from their base file. Also: an attempt to submit a file that is not an edit of the head revision of the file in the depot; typically occurs when another user opens the file for edit after you have opened the file for edit.

file pattern

Helix server command line syntax that enables you to specify files using wildcards.

file repository

The master copy of all files; shared by all users. In Helix server, this is called the depot.

file revision

A specific version of a file within the depot. Each revision is assigned a number, in sequence. Any revision can be accessed in the depot by its revision number, for example: testfile#3.

file tree

All the subdirectories and files under a given root directory.

file type

An attribute that determines how stores and diffs a particular file. Examples of file types are text and binary.

fix

A job that has been linked to a changelist.

form

Screens displayed by certain Helix server commands. For example, you use the change form to enter comments about a particular changelist and to verify the affected files.

full-file storage

The method by which Helix server stores revisions of binary files in the depot: every file revision is stored in full. Contrast this with reverse delta storage, which Helix server uses for text files.

G

get

An obsolete Helix server term. Replaced by sync.

Git Fusion

Git Fusion integrates Git with Helix, offering enterprise-ready Git repository management, and workflows that allow Git and Helix users to collaborate on the same projects using their preferred tools.

group

A list of Helix server users.

Groups

Groups is a feature of Helix Core that makes it easier to manage permissions for users.

H

have list

The list of file revisions currently in the client workspace.

head revision

The most recent revision of a file within the depot. Because file revisions are numbered sequentially, this revision is the highest-numbered revision of that file.

Helix server

The Helix Server depot and metadata; also, the program that manages the depot and metadata, also called Helix Versioning Engine.

HTTP

Hypertext Transfer Protocol, the communication protocol used to transfer or exchange hypertext, i.e. web pages.

I

iconv

iconv is a PHP extension that performs character set conversion, and is an interface to the GNU libiconv library.

integrate

To compare two sets of files (for example, two codeline branches) and determine which changes in one set apply to the other; determine if the changes have already been propagated; propagate any outstanding changes.

inter-file branching

Branching mechanism in Helix server.

J

job

A user-defined unit of work tracked by Helix server. The job template determines what information is tracked. The template can be modified by the Helix Server system administrator.

Job

A job is a component of the Helix Versioning Engine's defect tracking system and describes work to be done, such as a bug fix or improvement request. Associating a job with a changelist records which changes fixed the reported problem or added the requested improvement.

Job Daemon

A job daemon is a program that checks the Helix Core machine daily to determine if any jobs are open. If so, the daemon sends an email message to interested users, informing them the number of jobs in each category, the severity of each job, and more.

job specification

A specification containing the fields and valid values stored for a Helix server job.

job view

A syntax used for searching Helix server jobs.

jobspec

A job specification is a template describing the fields and possible values for each job stored in the Helix Core machine.

journal

A file containing a record of every change made to the Helix server's metadata since the time of the last checkpoint.

journaling

The process of recording changes made to the Helix server's metadata.

L

label

A named list of user-specified file revisions.

label view

The view that specifies which filenames in the depot can be stored in a particular label.

lazy copy

A method used by Helix server to make internal copies of files without duplicating file content in the depot. Lazy copies minimize the consumption of disk space by storing references to the original file instead of copies of the file.

license file

Ensures that the number of Helix server users on your site does not exceed the number for which you have paid.

list access

A protection level that enables you to run reporting commands but prevents access to the contents of files.

local depot

Any depot located on the currently specified Helix server.

local syntax

The operating-system-specific syntax for specifying a filename.

lock

A file lock prevents other clients from submitting the locked file. Files are unlocked with the p4 unlock command or submitting the changelist that contains the locked file.

log

Error output from the Helix server. By default, error output is written to standard error. To specify a log file, set the P4LOG environment variable or use the p4d -L flag when starting the service.

M

mapping

A single line in a view, consisting of a left side and a right side that specify the correspondences between files in the depot and files in a client, label, or branch. The left side specifies the depot files, and the right side specifies the client files. See also workspace view, branch view, label view.

MDS checksum

The method used by Helix server to verify the integrity of archived files.

merge

To create new files from existing files, preserving their ancestry (branching), or to propagate changes from one set of files to another. Also, the process of combining the contents of two conflicting file revisions into a single file, typically using a merge tool like P4Merge.

merge file

A file generated by the Helix server from two conflicting file revisions.

metadata

The data stored by the Helix server that describes the files in the depot, the current state of client workspaces, protections, users, labels, and branches. Metadata includes all the data stored in the service except for the actual contents of the files.

modification time

The time a file was last changed.

MPM

An MPM, or multi-processing module, is a component of the Apache web server that is responsible for binding to network ports, accepting requests, and dispatch operations to handle the request.

N

nonexistent revision

A completely empty revision of any file. Syncing to a nonexistent revision of a file removes it from your workspace. An empty file revision created by deleting a file and the #none revision specifier are examples of nonexistent file revisions.

numbered changelist

A pending changelist to which Helix server has assigned a number.

O

open file

A file that you are changing in your client workspace.

owner

The Helix server user who created a particular client, branch, or label.

P

p4

The Helix server command line program, and the command you issue to execute commands from the operating system command line.

p4d

The program that runs the Helix server; p4d manages depot files and metadata.

P4Diff

A Helix server application that displays the differences between two files. P4Diff is the default application used to compare files during the file resolution process.

P4PHP

P4PHP is the PHP interface to the Helix API, which enables you to write PHP code that interacts with a Helix Core machine.

PECL

PHP Extension Community Library, a library of extensions that can be added to PHP to improve and extend its functionality.

pending changelist

A changelist that has not been submitted.

Perforce

Perforce Software, provider of enterprise versioning tools.

PHP

PHP, or PHP: Hypertext Pre-processor, is an HTML-embedded scripting language designed to allow web developers to write dynamically generated web pages quickly.

Project

A project in Swarm is a group of Perforce users who are working together on a specific codebase, defined by one or more branches of code, along with options for a job filter, automated test integration, and automated deployment.

protections

The permissions stored in the Helix server's protections table.

PSD

Photoshop Document, a graphics file format, commonly with the filename extension .psd.

R

RCS format

Revision Control System format. Used for storing revisions of text files. RCS format uses reverse delta encoding for file storage. Helix server uses RCS format to store text files. See also reverse delta storage.

read access

A protection level that enables you to read the contents of files managed by Helix server.

remote depot

A depot located on a host other than that hosting the currently-specified Helix server.

Remote depot

A remote depot is a depot that acts like a pointer to a depot on a second Helix Core machine.

reresolve

The process of resolving a file after the file is resolved and before it is submitted.

resolve

The process you use to reconcile the differences between two revisions of a file. You can choose to resolve conflicts by selecting a file to be submitted or by merging the contents of conflicting files.

resource fork

One fork of a Mac file. (These files are composed of a resource fork and a data fork.) You can store resource forks in depots in the Helix server as part of an AppleSingle file by using the Helix Server server apple file type.

reverse delta storage

The method that Helix server uses to store revisions of text files. Helix server stores the changes between each revision and its previous revision, plus the full text of the head revision.

revert

To discard the changes you have made to a file in the client workspace.

review access

A special protections level that includes read and list accesses and grants permission to run the p4 review command.

review daemon

Any daemon process that uses the p4 review command. See also change review.

Review Daemon

A review daemon is a program that periodically checks the Helix Core machine to determine if any changelists have been submitted. If so, the daemon sends an email message to users who have subscribed to any of the files included in those changelists, informing them of changes in files they are interested in.

revision number

A number indicating which revision of the file is being referred to.

revision range

A range of revision numbers for a specified file, specified as the low and high end of the range. For example, `myfile#5,7` specifies revisions 5 through 7 of `myfile`.

revision specification

A suffix to a filename that specifies a particular revision of that file. Revision specifiers can be revision numbers, change numbers, label names, date/time specifications, or client names.

RPM

RPM Package Manager is a tool, and package format, for managing the installation, updates, and removal of software packages for Linux distributions such as Red Hat Enterprise Linux, the Fedora Project, and the CentOS Project.

S

server root

The directory in which `p4d` stores its metadata and all the shared files. To specify the server root, set the `P4ROOT` environment variable.

service

In the Helix Versioning Engine, the shared versioning service that responds to requests from Helix Server client applications. The Helix server (`p4d`) maintains depot files and metadata describing the files and also tracks the state of client workspaces.

shelving

The process of temporarily storing files in the Helix server without checking in a changelist.

status

For a changelist, a value that indicates whether the changelist is new, pending, or submitted. For a job, a value that indicates whether the job is open, closed, or suspended. You can customize job statuses.

stream

A stream is a branch, but with additional intelligence that determines what changes can be made and in what order they must be made.

submit

To send a pending changelist and changed files to the Helix server for processing.

subscribe

To register to receive email whenever changelists that affect particular files are submitted.

super access

An access level that gives the user permission to run every Helix server command, including commands that set protections, install triggers, or shut down the service for maintenance.

symlink file type

A Helix server file type assigned to symbolic links. On platforms that do not support symbolic links, symlink files appear as small text files.

sync

To copy a file revision (or set of file revisions) from the depot to a client workspace.

T

target file

The file that receives the changes from the donor file when you are integrating changes between a branched codeline and the original codeline.

text file type

Helix server file type assigned to a file that contains only ASCII text. See also binary file type.

TGA

Truevision Graphics Adapter, a graphics file format, commonly with the filename extension .tga.

theirs

The revision in the depot with which the client file is merged when you resolve a file conflict. When you are working with branched files, theirs is the donor file.

three-way merge

The process of combining three file revisions. During a three-way merge, you can identify where conflicting changes have occurred and specify how you want to resolve the conflicts.

TIFF

Tagged Image File Format, a graphics file format, commonly with the filename extension .tif or .tiff.

tip revision

In Helix server, the head revision. Tip revision is a term used by some other versioning systems.

trigger

A script automatically invoked by the Helix server when changelists are submitted.

two-way merge

The process of combining two file revisions. In a two-way merge, you can see differences between the files but cannot see conflicts.

typemap

A table in Helix server in which you assign file types to files.

U

user

The identifier that Helix server uses to determine who is performing an operation.

V

view

A description of the relationship between two sets of files. See workspace view, label view, branch view.

W

wildcard

A special character used to match other characters in strings. The following wildcards are available in Helix server: * matches anything except a slash; ... matches anything including slashes; %%0 through %%9 is used for parameter substitution in views.

workspace

See client workspace.

workspace view

A set of mappings that specifies the correspondence between file locations in the depot and the client workspace.

write access

A protection level that enables you to run commands that alter the contents of files in the depot. write access includes read and list accesses.

X

XSS

Cross-Site Scripting, a form of web-based attack that injects malicious code into a user's web browser.

Y

yours

The edited version of a file in the client workspace when you resolve a file. Also, the target file when you integrate a branched file.

License statements

Perforce Software includes software developed by the University of California, Berkeley and its contributors. This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>).