HelixCore

P4V User Guide

2020.1

April 2020

PERFORCE

www.perforce.com

Contents

What's new in this guide by release 10
- 2020.1 10
  - Large changelist area within dialog box 10
  - Graph Depot features 10
  - Edit recent workspaces 10
  - Custom page before and after a submit (HTML Actions) 10
  - Command-line enhancements 10
  - Stream spec enhancements 10
  - P4Admin features 11
- 2019.2 11
- 2019.1 11

How to use this guide 13
- Syntax conventions 13
- Feedback 13
- Other documentation 14

What's new in this guide by release 15
- 2020.1 15
  - Large changelist area within dialog box 15
  - Graph Depot features 15
  - Edit recent workspaces 15
  - Custom page before and after a submit (HTML Actions) 15
  - Command-line enhancements 15
  - Stream spec enhancements 15
  - P4Admin features 16
- 2019.2 16
- 2019.1 16

1 | Introduction 18
- About P4V, the Helix Visual Client 18
  - Basic concepts 18
- Getting started with P4V 19
- Checking for updates 19

2 | Using P4V 20
- Connecting to Helix server 20

Favorite connections 23
Connecting to a unicode-mode Helix server 23
Creating and managing workspaces 23
Create a workspace 24
Switch workspaces 29
Edit recent workspaces 31
View workspaces 32
Search for workspaces 32
Delete and unload workspaces 32
Navigating P4V 33
Terminology 33
Layout 34
Update status information 35
Modify views 35
Using access keys, shortcut keys, and drag-and-drop shortcuts 36
Access keys 36
Shortcut keys 36
Drag-and-drop shortcuts 40
About P4V icons 41
Bookmarking files 43
Bookmark a file or folder 44
Manage bookmarks 44
Searching and filtering 45
Find files in a depot or workspace 46
Find a file, folder, or item in the active tab 47
Find specs using filters 47
Customize Depot and Workspace views using filters 48
Save and reuse filters 50
Filter with file paths 52
Reconciling offline work 54
Exporting files 56
Formatting text in Description fields 57
3 | Configuring P4V 58
Configuring P4V preferences 58
Connections preferences 59
Streams preferences 59

Server Data preferences 61
Behavior preferences 64
Double-click preferences 70
Shortcuts preferences 70
Logging preferences 70
Display preferences 71
Files and History preferences 72
Fonts preferences 72
Features preferences 73
Tools preferences 73
File Editors preferences 74
Diff preferences 74
Merge preferences 75
HTML Tools preferences 76
Editing user preferences 77
Viewing effective settings 79
4 | Managing files 80
Adding files to the depot 80
Retrieving files from the depot 81
Editing files 82
Reverting files 82
Checking in files 84
View changelists 84
Submit changelists 84
Reverse a changelist submission 85
Restrict access to a changelist 86
Configure changelist display 86
Undoing changes 87
Displaying revision history 88
Files 89
Folders 89
Changelists 90
Changing a file’s type 90
Renaming and moving files or folders 91
Cleaning up files and directories 91
Deleting files 92

Diffing files and folders 92
Diff dialog options 93
Check workspace consistency 94
View the state of the depot at a specified point in time 94
Diff large files 94
Shelving files 95
Shelve checked-out files in a pending changelist 96
Unshelve files 96
Submit shelved files 97
Delete shelved files 98
5 | More file management tools 99
Viewing codeline history in the revision graph 99
Read the revision graph 99
Navigate the revision graph 100
Filter the revision graph 101
Display details 101
Viewing file history with Time-lapse View 101
Toolbar 102
Slider 103
Viewing image file history with Time-lapse View 104
Using the folder diff utility 105
Diffing text files 106
Diffing images 107
Merging files 108
6 | Managing codelines 109
Creating branches 109
Merging files between codelines 110
Open files for merge 111
Resolving files 113
Resolve individual files 113
Resolve multiple files 116
Managing branch mapping 118
Work with branch mapping 118
Managing labels 120
Create labels 120

Label files 121
Display and search for labels 121
Edit labels 122
Delete and unload labels 122
Retrieve file revisions in a label 123
Display files associated with a label 123
Managing jobs 123
Create a job 124
Add a job to a pending changelist 124
View jobs 124
Search for jobs 124
7 | Advanced P4V options 130
Configuring custom tools 130
Environment considerations 130
Add custom tools 131
Import and export tools 133
HTML Actions 134
Run the example of pre and post submit 134
P4VJS and the examples 135
P4VJS functions for Submit 136
Replacing the Submit dialog is possible 136
To develop your own custom HTML Action 137
Launching P4V components from the command-line client, P4VC 137
Command Syntax 137
8 | Working with streams 141
About streams 141
The mainline model 142
Stream views 142
Propagating change between streams 142
Stream depots 143
Terminology differences between command line client and P4V 144
About stream views 144
Stream path types 145
Stream path syntax 145
Inheritance between parents and children 146
Examples 147

Setting up streams 150
Creating streams 151
Creating stream workspaces 154
Editing streams 154
Edit a stream publicly 155
Edit a stream privately 155
Resolving streams 157
Shelving streams 158
Streams Revision Graph 159
Selecting streams 161
Using the stream graph 161
Stream Graph display conventions 161
Configure the stream graph display 163
Display stream status 164
Work in a stream 164
Branching with Streams 165
Merging down and copying up between streams 165
Merge down 166
Copy up 167
Propagate change between unrelated streams 167
Deleting stream files and streams 167
Working with task streams 168
Overview 168
Create a task stream from a parent stream in the same depot 169
Create a task stream in a different depot 170
Create a task stream without a parent 171
Convert a task stream to a regular stream 172
Delete and unload task streams 172
Filter task stream files out of File History results 173
Working with virtual streams 173
View virtual streams 174
Stream path behavior in virtual streams 175
Create a virtual stream 176
Submit changes to a virtual stream 176
Merge and copy to a virtual stream 177
9 | Working with graph depots 178

Repo folder information 179
Repo file information 180
10 | Using P4V for distributed versioning 183
Understanding DVCS and setting up the server 183
Init 183
Read this first 184
Directories and files 184
Add files 185
Clone 185
Prepare to fetch and push content between servers 186
Fetch and push 186
Configure security for fetching and pushing 187
Specify what to copy 187
What do Fetch and Push copy? 187
Resubmit 187
Branches 188
Understanding remote mappings 189
11 | Integration with Swarm 191
Review workflow 191
Setting up the Swarm integration 192
Swarm integration features 192
Request a review 193
Update a review 196
Open a review in Swarm 198
Review ID and Review State columns 198
Reconnect to Swarm 199
Glossary 200
License statements 219

What's new in this guide by release

2020.1

Large changelist area within dialog box

If the number of files in the changelist exceeds the limit of **Edit > Preferences > Server Data >** "Maximum number of files displayed per changelist " on page 62, the dialogs for **Submit**, **Shelve**, **Unshelve**, **Revert**, and **Resolve** display the files differently. See "Server Data preferences" on page 61.

Graph Depot features

Users can create, open, and sync hybrid workspaces to browse and view both Perforce and Git (graph depot) source files. See "Working with graph depots" on page 178.

Edit recent workspaces

You can specify how many recent workspaces are listed and remove the recent workspaces you no longer want. See "Edit recent workspaces" on page 31.

Custom page before and after a submit (HTML Actions)

You can make P4V present a custom HTML page before and after the user submits a changelist. See "HTML Actions" on page 134.

Command-line enhancements

Although P4V is primarily used a graphic user interface, it does have a command-line interface. The new commands are **`p4vc history`**, **`p4vc properties`**, and **`p4vc diffprev`**. In addition, **`p4vc submit`** now supports launching with a file names. See "Launching P4V components from the command-line client, P4VC" on page 137.

Stream spec enhancements

- displays any custom fields of the stream spec form. (Such fields are created on the server command line by using the p4 streamspec command.)
- supports **`diff stream against previous`**

P4Admin features

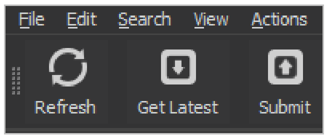
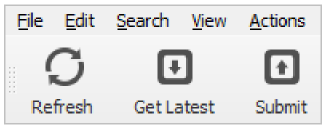
- creating and viewing depots of type `graph`
- setting access control for stream specs (see Managing Permissions in *P4Admin User Guide*)

2019.2

For a complete list about the product, see the 2019.2 *P4V Release Notes*. New information in this Guide is at:

- "Streams Revision Graph" on page 159
- "Shelving streams" on page 158
- Support for P4V connecting to a Helix Core server that uses Helix Authentication Service - see the second Note under Step 5 at "Connecting to Helix server" on page 20

"Display preferences" on page 71 allow the choice of:

Dark theme for a dark background with light font:	
or the default Light theme:	

2019.1

Following is a summary of new information with links to topics. For a complete list, see 2019.1 *P4V Release Notes*.

- Introduced private editing of streams. This feature lets you modify a stream in isolation from other users of that stream, as opposed to having changes become global as soon as you save them. Privately edited streams get checked out and added to a changelist, allowing for advanced testing and enhanced traceability. For details, see "Editing streams" on page 154 and "Resolving streams" on page 157.
- Added the option to filter the depot view by stream type. For details, see "Customize Depot and Workspace views using filters" on page 48.
- When switching streams in the current workspace, P4V now:
  - Offers the option to perform a reconcile operation when switching streams in the current workspace. For details, see "Streams preferences" on page 59.

- Automatically shelves files checked out to the default changelist. For details, see "Work in a stream" on page 164
- Prompts you to shelve files checked out to a numbered changelist.

For details, see "Streams preferences" on page 59 and "Work in a stream" on page 164.

How to use this guide

P4V, the Helix Visual Client, is the cross-platform graphical user interface for the Helix Core server. To use the Helix Core server to manage files, you typically connect to it using an application like P4V. P4V enables you to check files in and out, and perform various other versioning tasks.

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.

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.
italics	A parameter for which you must supply specific information. For example, for a *serverid* parameter, supply the ID of the server.
[`-f`]	The enclosed elements are optional. Omit the brackets when you compose the command.
`...`	Previous argument can be repeated. ▪ `p4 [g-opts] streamlog [ -l -L -t -m max ]` ***`stream1`*** `...` means `1` or more stream arguments separated by a space ▪ See also the use on `...` in Command alias syntax in the *Helix Core P4 Command Reference* **Tip** `...` has a different meaning for directories. See Wildcards in the *Helix Core P4 Command Reference*.
element1 | *element2*	Either *element1* or *element2* is required.

Feedback

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

Other documentation

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

Tip
You can also search for Support articles in the Perforce Knowledgebase.

What's new in this guide by release

2020.1

Large changelist area within dialog box

If the number of files in the changelist exceeds the limit of **Edit > Preferences > Server Data >** "Maximum number of files displayed per changelist " on page 62, the dialogs for **Submit**, **Shelve**, **Unshelve**, **Revert**, and **Resolve** display the files differently. See "Server Data preferences" on page 61.

Graph Depot features

Users can create, open, and sync hybrid workspaces to browse and view both Perforce and Git (graph depot) source files. See "Working with graph depots" on page 178.

Edit recent workspaces

You can specify how many recent workspaces are listed and remove the recent workspaces you no longer want. See "Edit recent workspaces" on page 31.

Custom page before and after a submit (HTML Actions)

You can make P4V present a custom HTML page before and after the user submits a changelist. See "HTML Actions" on page 134.

Command-line enhancements

Although P4V is primarily used a graphic user interface, it does have a command-line interface. The new commands are **`p4vc history`**, **`p4vc properties`**, and **`p4vc diffprev`**. In addition, **`p4vc submit`** now supports launching with a file names. See "Launching P4V components from the command-line client, P4VC" on page 137.

Stream spec enhancements

- displays any custom fields of the stream spec form. (Such fields are created on the server command line by using the p4 streamspec command.)
- supports **`diff stream against previous`**

P4Admin features

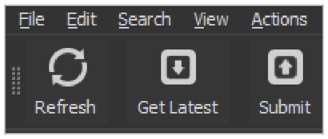
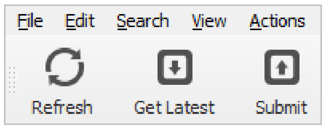
- creating and viewing depots of type `graph`
- setting access control for stream specs (see Managing Permissions in *P4Admin User Guide*)

2019.2

For a complete list about the product, see the 2019.2 *P4V Release Notes*. New information in this Guide is at:

- "Streams Revision Graph" on page 159
- "Shelving streams" on page 158
- Support for P4V connecting to a Helix Core server that uses Helix Authentication Service - see the second Note under Step 5 at "Connecting to Helix server" on page 20

"Display preferences" on page 71 allow the choice of:

Dark theme for a dark background with light font:	 File Edit Search View Actions Refresh Get Latest Submit
or the default Light theme:	 File Edit Search View Actions Refresh Get Latest Submit

2019.1

Following is a summary of new information with links to topics. For a complete list, see 2019.1 *P4V Release Notes*.

- Introduced private editing of streams. This feature lets you modify a stream in isolation from other users of that stream, as opposed to having changes become global as soon as you save them. Privately edited streams get checked out and added to a changelist, allowing for advanced testing and enhanced traceability. For details, see "Editing streams" on page 154 and "Resolving streams" on page 157.
- Added the option to filter the depot view by stream type. For details, see "Customize Depot and Workspace views using filters" on page 48.
- When switching streams in the current workspace, P4V now:
  - Offers the option to perform a reconcile operation when switching streams in the current workspace. For details, see "Streams preferences" on page 59.

- Automatically shelves files checked out to the default changelist. For details, see "Work in a stream" on page 164
- Prompts you to shelve files checked out to a numbered changelist.

For details, see "Streams preferences" on page 59 and "Work in a stream" on page 164.

1 | Introduction

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

About P4V, the Helix Visual Client 18
Basic concepts 18
Getting started with P4V 19
Checking for updates 19

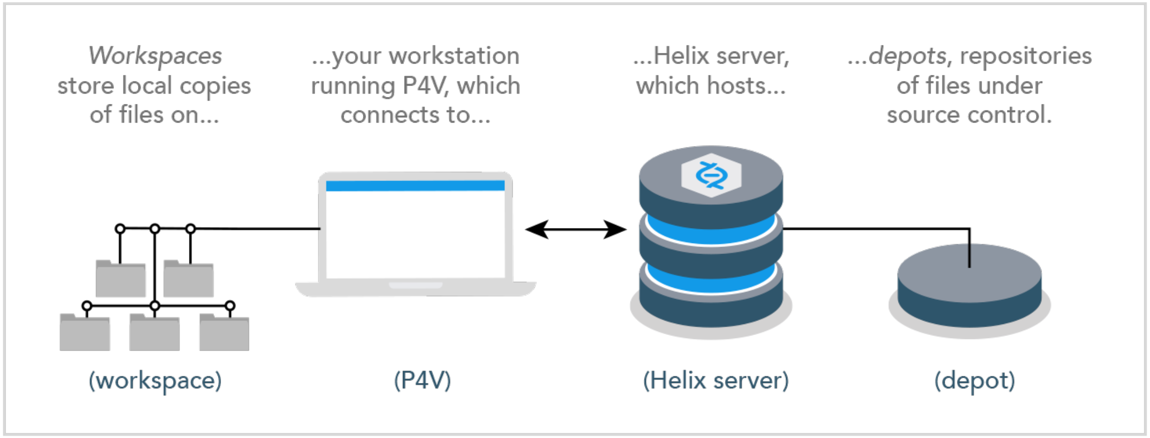
About P4V, the Helix Visual Client

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

Basic concepts

The Helix Core server 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 the Helix Core server 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.

Helix Core server users connect to a shared file repository using a client application like P4V. P4V connects your computer to Helix server and helps you move files between the Helix server depots and your workspace, as illustrated in the following figure.


Workspaces store local copies of files on...
...your workstation running P4V, which connects to...
...Helix server, which hosts...
...*depots*, repositories of files under source control.
(workspace)
(P4V)
(Helix server)
(depot)

The definitions for these terms are as follows:

- **Workspace**: folders or directories on your workstation where you work on revisions of files that are managed by the Helix Core server.
- **Helix Core app**: P4V (or another Helix Core application, like the command-line client or P4VS, the Helix Plugin for Visual Studio), running on your workstation, which makes requests from the Helix Core server and delivers the results of those requests (files, status information, and so on) to you.
- **Helix Core server** or **Helix server**: the program that responds to requests from Helix Core applications, maintains depot files, and tracks the state of workspaces.
- **Depot**: a file repository hosted by the Helix server. It contains all existing versions of all files ever submitted. The Helix server 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 Helix server instance (see "Connecting to Helix server" on the facing page)
- Configure your client workspace (see "Creating and managing workspaces" on page 23)
- Get files from the depot (see "Retrieving files from the depot" on page 81)
- Add files to the depot (see "Adding files to the depot" on page 80)

For short videos on basic and more complex operations in P4V, check out the video library on the Perforce website.

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 might be disabled by your Helix Core 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 20
- Favorite connections 23
- Connecting to a unicode-mode Helix server 23

Creating and managing workspaces 23
- Create a workspace 24
- Switch workspaces 29
- Edit recent workspaces 31
- View workspaces 32
- Search for workspaces 32
- Delete and unload workspaces 32

Navigating P4V 33
- Terminology 33
- Layout 34
- Update status information 35
- Modify views 35

Using access keys, shortcut keys, and drag-and-drop shortcuts 36
- Access keys 36
- Shortcut keys 36
- Drag-and-drop shortcuts 40

About P4V icons 41

Bookmarking files 43
- Bookmark a file or folder 44
- Manage bookmarks 44

Searching and filtering 45
- Find files in a depot or workspace 46
- Find a file, folder, or item in the active tab 47
- Find specs using filters 47
- Customize Depot and Workspace views using filters 48
- Save and reuse filters 50
- Filter with file paths 52

Reconciling offline work 54

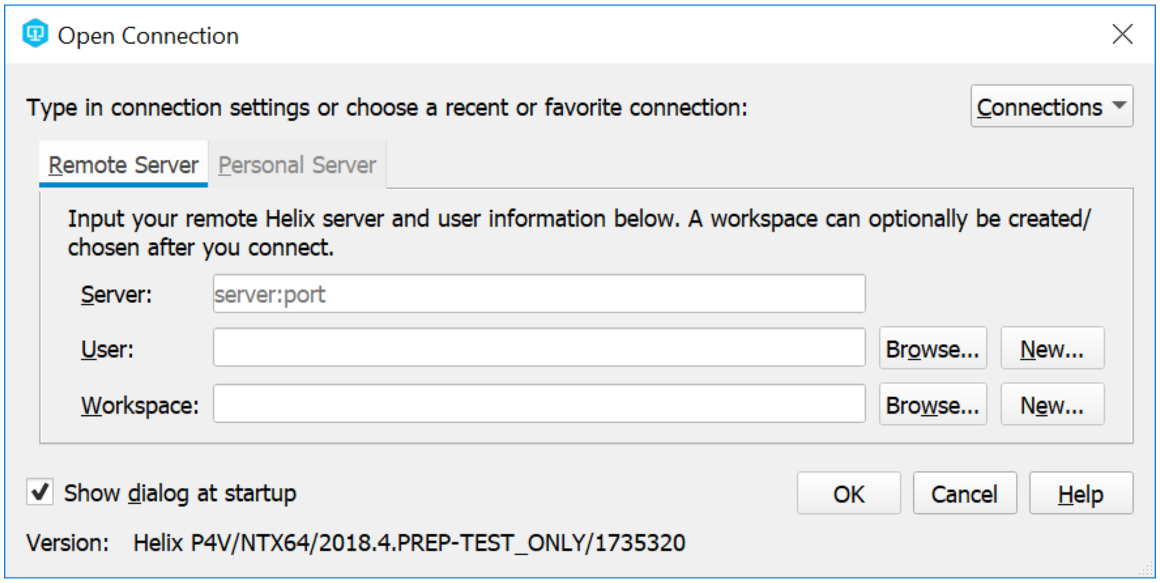
Exporting files 56

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 Helix server using a new connection:

1. Launch P4V or, if P4V is already running, go to **Connection > Open Connection**. The **Open Connections** dialog opens.



Open Connection

Type in connection settings or choose a recent or favorite connection:

Input your remote Helix server and user information below. A workspace can optionally be created/chosen after you connect.

Server: server:port

User:

Workspace:

✓ Show dialog at startup

Version: Helix P4V/NTX64/2018.4.PREP-TEST_ONLY/1735320

2. In the **Open Connection** dialog, enter the Helix server name and port number for the connection using the following syntax: `server_host:port_number`

   If your Helix server 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 Helix server 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. For details, see "Create a workspace" on page 24.
5. Click **OK**.

   P4V connects to the specified Helix server and displays a new instance of its main window.

> **Note**
> If the server you are connecting to is configured with multi-factor authentication (MFA), you are prompted for another layer of verification. Depending on the setup, you may need to select a method of verification before you can enter your credentials.
>
> For more information, see `p4 login2` in the *Helix Core P4 Command Reference*.

> **Note**
> If the server you are connecting to is configured for authentication with Helix Authentication Service, the Identity Provider (IdP) web page opens, prompting you for the credentials registered with your Identity Provider (IdP). For details, see https://github.com/perforce/helix-authentication-service or contact your Helix Core server administrator.

> **Note**
> If the server you are connecting to is configured for authentication with Helix SAML, the Helix SAML dialog opens, prompting you for the user name and password registered with your Identity Provider (IdP). For details, see "Helix SAML" in the in the *Helix Core Server Administrator Guide* or contact your Helix Core server administrator.

To connect to Helix server 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.

> **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 File 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**. 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 Helix server 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** and select the connection you want to open.

Connecting to a unicode-mode Helix server

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

If you are connecting to a unicode-mode Helix server 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 Helix server.

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

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 Helix server instances 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

Your Helix server administrator can set up your workspace for you, but it is advisable to learn a few important aspects of configuring your workspace, specifically configuring the workspace root directory and configuring the workspace specification, also called the view.

The *workspace root*, also referred to as client root, specifies the location on your workstation under which Helix server stores copies of depot files. You should give it a meaningful name and make sure it is not set to your computer's root directory.

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. Users of P4V call this location the *workspace*. (Users of the he Helix Core server command line call this location the *client*.)

A computer can contain multiple workspaces. Although P4V can connect to the server using only host name and port, a workspace is required to work with files that the Helix server manages.

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 Helix server, based on the structure of the stream. If the structure of the stream changes, the workspace view is updated automatically. You cannot manually edit a stream's workspace view. If you use a classic depot, a graph depot, or a hybrid workspace, you must define and maintain the workspace view manually.

Note

- Helix server streams provide structured branching for version control of related files, such as codelines. Stream affect the way you create depots, define client workspaces, and integrate changes between files. See "Working with streams" on page 141.
- For information about graph depots and hybrid workspaces, see "Working with graph depots" on page 178.

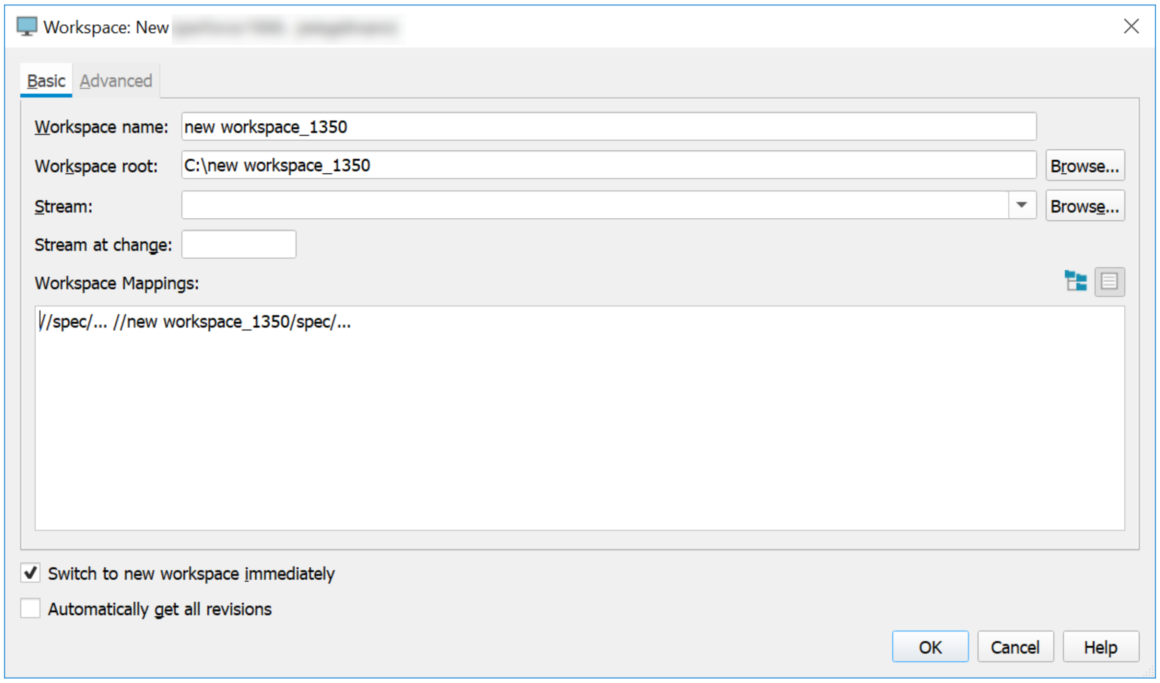
Create a workspace

When you create a new workspace, P4V immediately creates the workspace directory. (In versions prior to 2019.1, P4V would only create the directory when you synced files for the first time.)

To create a new workspace:

1. Use of the following three alternatives:
   - On the top menu, choose **Connection > New Workspace**.
   - In the **Streams** tab (in the right pane), right-click a stream and select **New Workspace**.
   - In the **Workspaces** tab (in the right pane), right-click and select **New Workspace**.
2. If P4V is configured to prompt for a new workspace name, the **Workspace Name** dialog appears. Enter a name and click **OK**.

   Otherwise, in the **Workspace** dialog, on the **Basic** tab, accept or change the default workspace name and root.


Workspace: New [redacted]

Basic | Advanced

Workspace name: new workspace_1350

Workspace root: C:\new workspace_1350 Browse...

Stream: Browse...

Stream at change:

Workspace Mappings:

```
//spec/... //new workspace_1350/spec/...
```

☑ Switch to new workspace immediately

☐ Automatically get all revisions

OK Cancel Help

3. Map the workspace to the depot:

   Classic depots (free-form branching):

   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 ✗☰ buttons above the field.
     Alternatively, right-click and select the **Include**, **Exclude**, and **Clear** options in the context menu.

   For more information about mapping workspaces, including an example, see "Define a workspace view" on page 28.

   Stream depots (structured branching):

   - **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 were 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 Core Server User Guide*.

4. Specify the following settings on the **Advanced** tab as needed.

Owner	The user who created the specification.
Locked	If selected, only the owner of the workspace can use, change, or delete the workspace specification. Other users can see the workspace, but they cannot edit or delete its spec or use it to sync, open files, or do anything else.
Description	Your own explanation of the purpose of the workspace, or any related information you need to specify. The **Description** field accepts HTML tags for marking up and hyperlinking text. For details, see "Formatting text in Description fields" on page 57.
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	■ **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 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. With this option, P4V also minimizes costly digest computations on the client by checking file modification times before checking digests to determine if files have been modified outside of P4V. ■ **Rmdir**: Deletes a workspace folder if all the files contained in the folder are removed.
Line ending characters for text files	The line-end convention used for storing text files on the workspace computer: ■ **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: ■ **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	If selected, P4V reopens submitted files in the default changelist.

Client type	Specifies the type of client: `writeable` — The default. `readonly` — for short lived clients used in build automation scripts. Such clients cannot edit or submit files. `partitioned` — similar to `readonly` but with the additional ability to edit and submit files using that client. **Note** Using writeable clients in build automation scripts can fragment the `db.have` table, which records the files that a client has synced. If you are experiencing performance issues when syncing, consider using use a read-only or partitioned client. A client of type `readonly` or `partitioned` is assigned its own `db.have` table. The location of this table must first be specified with the `client.readonly.dir` configurable by an administrator. See also "Using read-only and partitioned clients in automated builds" in *Helix Core Server Administrator Guide*
Backup	Not currently in use. Applies only to clients bound to cluster workspace servers. Server clustering is no longer supported in Helix server. For additional information, see the Perforce Server Clustering Update.

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

Define a workspace view

The workspace view determines which portions of the depot are visible in your **Workspace** tab (in the left pane) 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 here.

To define or change your workspace view:

1. If the **Workspace** form is not already open, do the following:
    a. Go to **View > Workspaces** to open the **Workspaces** tab.
    b. In the **Workspaces** tab, right-click the workspace and select **Edit Workspace**.
2. In the **Workspace** dialog, edit the **Workspace Mappings** field.

   You can define the view *syntactically* and *graphically*, as described in "Syntactic view specification" on the next page and "Graphical view specification" on the next page.
3. When you have finished editing, save your changes.

Syntactic view specification

In the **Workspace** dialog, on the **Basic** tab, above the **Workspace Mappings** field, 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. The following example illustrates this.

Note
The workspace path provided includes the workspace *name* (`bruno`), not the workspace *root* (`C:\Users\Perforce\bruno`).

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

For details about client view syntax, see the Configure clients chapter in the *Helix Core Server User Guide*.

Graphical view specification

Click the **View workspace mapping as tree** icon . The depot is displayed as a tree of folders and files. Right-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 depot syntax to specify the workspace view.
- Clear: Remove mapped folders or files from the mapping.

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 **Search > Filter Depot > Entire Depot Tree**, right-click the desired folder in the Depot Tree, and choose **Map to Workspace View**.

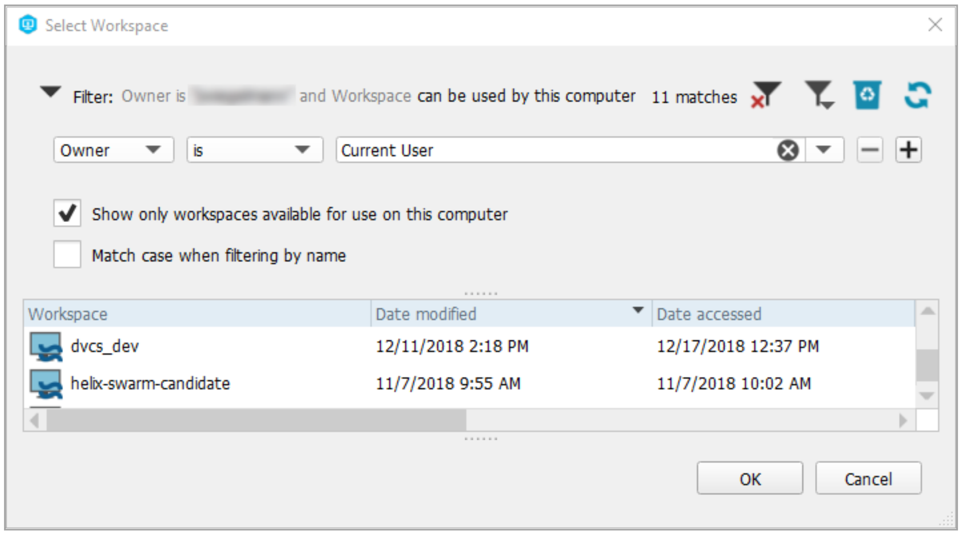
Switch workspaces

How to change from one workspace to another depends on whether you work with classic depots or streams.

Classic depots

When working with classic depots, you can switch your workspace using the **Select Workspace** dialog. To open the **Select Workspace** dialog, do any of the following:

- In the drop-down list 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.



Select Workspace

Filter: Owner is [redacted] and Workspace can be used by this computer 11 matches

Owner | is | Current User

✓ Show only workspaces available for use on this computer

Match case when filtering by name

Workspace	Date modified	Date accessed
dvcs_dev	12/11/2018 2:18 PM	12/17/2018 12:37 PM
helix-swarm-candidate	11/7/2018 9:55 AM	11/7/2018 10:02 AM

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

Stream depots

When working with streams, you can switch workspaces using the same methods you use when working with classic depots. In addition, you can switch stream workspaces by merging or copying to a stream (in this case, P4V prompts you to switch to the target stream's workspace), or by doing the following:

- In the **Streams** tab or the **Stream Graph**, right-click a stream and select **Work in this Stream**.
- In the **Stream Graph**, drag the Workspace icon from your current stream to the one you want to work in.

If your stream preferences are set to use a different workspace and to show an information dialog when switching workspaces, P4V prompts you to switch workspaces or create a new workspace. In this case, click the **Create New Workspace** button (if you have only one workspace), **Switch Workspaces** button (if you have two workspace), or the **Select Workspaces** button (if you have more than two workspace) 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.

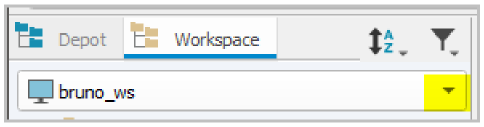
If your preferences are set to use the same workspace when you switch between streams, the workspace view changes to include the stream you are switching to. In other words, the **Stream** field value in the workspace definition changes to the new stream. When you do this and your Helix server version is 2019.1 or later, then if you have files checked out to:

- **The default changelist:** P4V shelves them automatically in a numbered changelist with the description "Switched branch shelf" and unshelves them again to the default changelist when you switch back to that stream. However, note that the numbered changelist created for the interim shelf does not get deleted and stays behind in the system.
- **A numbered changelist:**P4V prompts you to shelve those files before switching. When you switch back to that stream, those files remain shelved in the numbered changelist. You have to unshelve them manually.

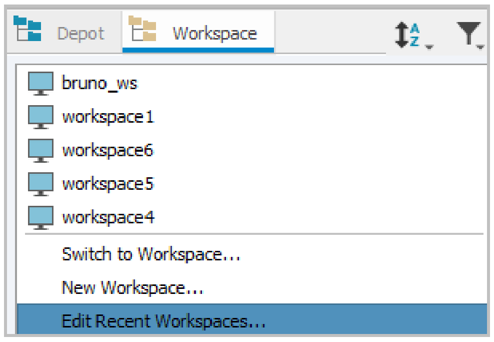
Note
If you reuse a workspace between streams and you have checked out the stream, you cannot switch streams. You first need to check in the stream. See also "Edit a stream privately" on page 155.

Edit recent workspaces

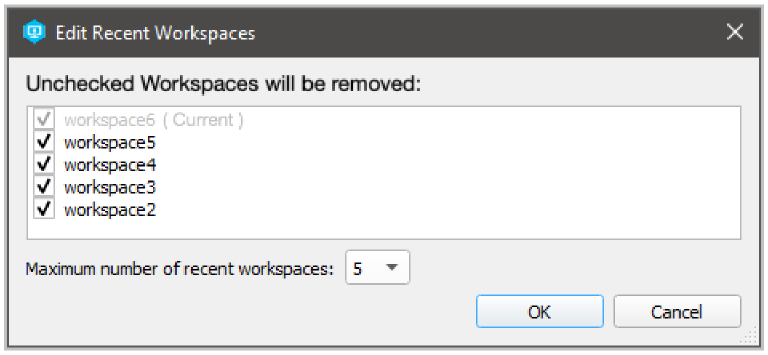
1. Open the workspace list:



2. Click **Edit Recent Workspaces...**



3. To remove a workspace, unselect its check box.



Unchecked Workspaces will be removed:

- ✓ workspace6 (Current)
- ✓ workspace5
- ✓ workspace4
- ✓ workspace3
- ✓ workspace2

Maximum number of recent workspaces: 5

Note that the current workspace cannot be removed.

To change the **Maximum number of recent workspaces** to display on the workspace list, select a value from the drop-down list.

View workspaces

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

- Go to **View > Workspaces**. Then double-click a workspace row in the **Workspaces** tab to display the details of the client workspace specification.
- Open the **Select Workspace** dialog (for details, see "Switch workspaces" on page 29). Then double-click a workspace row to display the details of the client workspace specification.

Search for workspaces

You can use filters to search for workspaces in 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 45.

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.

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 delete a workspace:

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

To unload a workspace:

1. Submit or revert any pending or shelved changelists associated with the workspace.
2. Go to **View > Workspaces** to open the **Workspaces** tab.
3. Right-click the workspace and select **Unload Workspace '*workspace_name*''**.

To reload an unloaded workspace:

1. Go to **View > Workspaces** 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. Right-click the workspace and select **Reload Workspace '*workspace_name*'**.

For more information about unloading, see the `p4 unload` command in the *Helix Core P4 Command Reference*.

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.

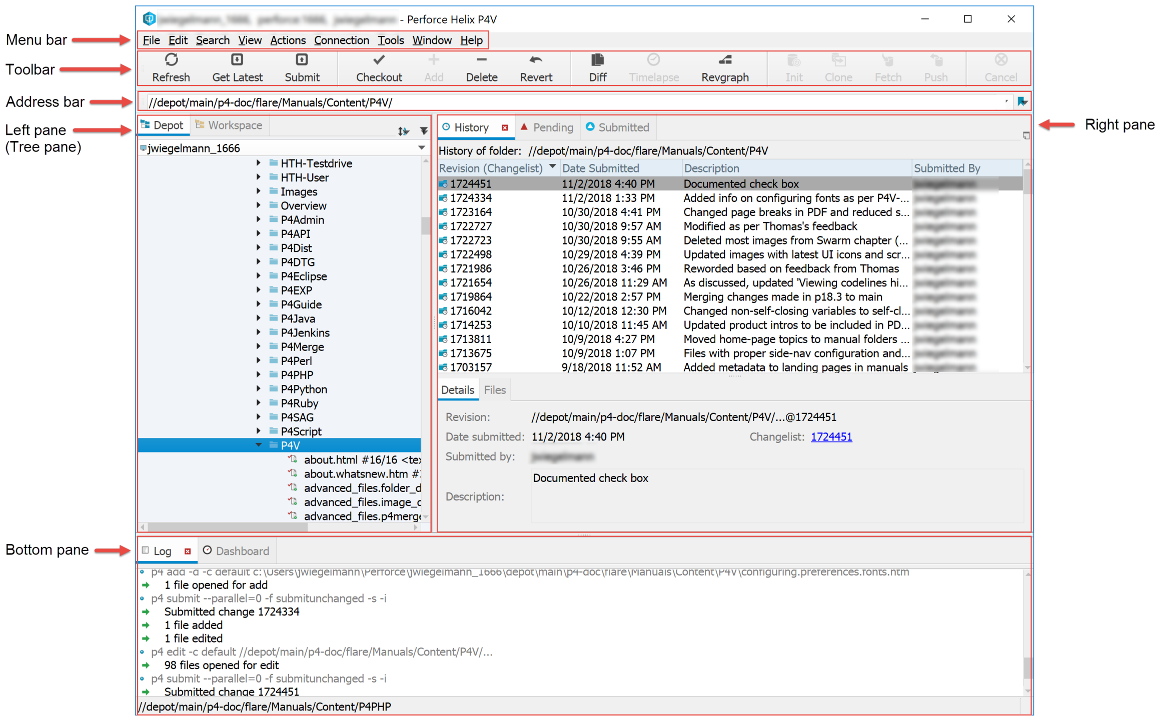
Terminology

P4V menus and forms use a general approach to versioning terminology and actions, to ensure that 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.

Helix server 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 three panes: the left pane, which is also called the **Tree** pane; the right pane, which is where you do most of your work in P4V; and the bottom pane, which displays log and dashboard information. The following figure illustrates the different elements in the P4V user interface. They are described in detail below.


Menu bar
Toolbar
Address bar
Left pane (Tree pane)
Bottom pane
Right pane
[redacted] - Perforce Helix P4V
File Edit Search View Actions Connection Tools Window Help
Refresh Get Latest Submit Checkout Add Delete Revert Diff Timelapse Revgraph Init Clone Fetch Push Cancel
Depot Workspace
History Pending Submitted
Details Files
Log Dashboard

//depot/main/p4-doc/flare/Manuals/Content/P4V/

jwiegelmann_1666
HTH-Testdrive, HTH-User, Images, Overview, P4Admin, P4API, P4Dist, P4DTG, P4Eclipse, P4EXP, P4Guide, P4Java, P4Jenkins, P4Merge, P4Perl, P4PHP, P4Python, P4Ruby, P4SAG, P4Script, P4V
about.html #16/16 <te
about.whatsnew.htm #
advanced_files.folder_d
advanced_files.image_c
advanced_files.p4merg

History of folder: //depot/main/p4-doc/flare/Manuals/Content/P4V

Revision (Changelist)	Date Submitted	Description	Submitted By
1724451	11/2/2018 4:40 PM	Documented check box	[redacted]
1724334	11/2/2018 1:33 PM	Added info on configuring fonts as per P4V-...	[redacted]
1723164	10/30/2018 4:41 PM	Changed page breaks in PDF and reduced s...	[redacted]
1722727	10/30/2018 9:57 AM	Modified as per Thomas's feedback	[redacted]
1722723	10/30/2018 9:55 AM	Deleted most images from Swarm chapter (...	[redacted]
1722498	10/29/2018 4:39 PM	Updated images with latest UI icons and scr...	[redacted]
1721986	10/26/2018 3:46 PM	Reworded based on feedback from Thomas	[redacted]
1721654	10/26/2018 11:29 AM	As discussed, updated 'Viewing codelines hi...	[redacted]
1719864	10/22/2018 2:57 PM	Merging changes made in p18.3 to main	[redacted]
1716042	10/12/2018 12:30 PM	Changed non-self-closing variables to self-cl...	[redacted]
1714253	10/10/2018 11:45 AM	Updated product intros to be included in PD...	[redacted]
1713811	10/9/2018 4:27 PM	Moved home-page topics to manual folders ...	[redacted]
1713675	10/9/2018 1:07 PM	Files with proper side-nav configuration and...	[redacted]
1703157	9/18/2018 11:52 AM	Added metadata to landing pages in manuals	[redacted]

Revision: //depot/main/p4-doc/flare/Manuals/Content/P4V/...@1724451
Date submitted: 11/2/2018 4:40 PM Changelist: 1724451
Submitted by: [redacted]
Description: Documented check box

```
p4 add -d -c default c:\Users\jwiegelmann\Perforce\jwiegelmann_1666\depot\main\p4-doc\flare\Manuals\Content\P4V\configuring.preferences.fonts.htm
    1 file opened for add
p4 submit --parallel=0 -f submitunchanged -s -i
    Submitted change 1724334
    1 file added
    1 file edited
p4 edit -c default //depot/main/p4-doc/flare/Manuals/Content/P4V/...
    98 files opened for edit
p4 submit --parallel=0 -f submitunchanged -s -i
    Submitted change 1724451
```

//depot/main/p4-doc/flare/Manuals/Content/P4PHP

- **Menu bar**: Provides access to all available options.
- **Toolbar**: Provides quick access to a subset of actions and tools available from the menu bar. To get information about a toolbar item or other object in P4V, move the mouse pointer over the object. P4V displays a small window (tooltip) containing status or explanatory information about the object. P4V makes extensive use of tooltips.
- **Address bar**: Lets you navigate to specific folders and files in the depot and in your workspace. You can copy from and paste into the address bar.
- **Left pane** (tree pane): Includes the following tabs:
  - **Depot Tree**: 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**.
  - **Workspace Tree**: Shows the files on your computer, including files that are *not* in the depot.
- **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**.

- **Bottom pane**: Includes the following tabs:
  - **Log tab**: Displays the commands issued by P4V. To display this pane, choose **View > Log**.
  - **Dashboard pane**: Displays details about the status of your workspace and provides quick links for common tasks.

    To display the dashboard, choose **View > Dashboard**. To configure the tasks displayed in the dashboard, click the Settings button in 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 "right-click" when instructing you to display a context menu.)

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 Helix server. To force a refresh operation, click .

Modify views

You can 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 right-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 43.
- **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 45.

- **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.

Using access keys, shortcut keys, and drag-and-drop shortcuts

You can use access keys, shortcut keys, and drag-and-drop shortcuts to work quickly and efficiently in P4V.

Access keys

On Windows, you can use access keys to open menus and navigate the user interface. An access key is an alphanumeric key that you can use instead of a mouse pointer to activate menu or dialog options. It correlates to a designated character in a control label. Access keys are associated only with controls that have text labels. In P4V, you can identify access keys by looking for underlined characters in labels.

To access a menu and to activate a control within a dialog box, you need to press **ALT+<letter>**. For example:

- To open the **File** menu, press **ALT+F**.
- To select the **Latest revision** option in the **Diff** dialog, press **ALT+L**.

Within a menu, you only need to press the **<key>** key. For example, to switch to the **Depot Tree** view:

1. Press **ALT+V** to open the **View** menu.
2. Press **D** to switch to the **Depot Tree** view (if your current selection is the **Workspace Tree**).

Shortcut keys

Shortcut keys allow more experienced users to quickly perform select actions. For example:

- To switch to the Depot Tree, press **Ctrl+9**.
- To switch to the Workspace Tree, press **Ctrl+0**.

Where applicable, P4V displays shortcut keys to the right of a menu label.

The following table lists common shortcut keys.

Action	Shortcut Key
Show in Explorer	Ctrl+Shift+S
Print	Ctrl+P
Undo	Ctrl+Z
Redo	Ctrl+Y
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Select All	Ctrl+A
Find	Ctrl+F
Find File	Ctrl+Shift+F
Go To	Ctrl+G
View Depot Tree	Ctrl+9
View Workspace Tree	Ctrl+0
View Pending Changelists	Ctrl+1
View Submitted Changelists	Ctrl+2
View Branch Mappings	Ctrl+3
View Labels	Ctrl+4
View Workspaces	Ctrl+5
View Jobs	Ctrl+6
View Streams	Ctrl+7
View Remote Mappings	Ctrl+8
Refresh All	F5
Get Latest Revision	Ctrl+Shift+G
Check Out	Ctrl+E
Submit	Ctrl+S
Revert	Ctrl+R
Initialize Personal Server	Ctrl+Shift+I

Action	Shortcut Key
Fetch	Ctrl+Shift+T
Push	Ctrl+Shift+P
Lock	Ctrl+Shift+L
Unlock	Ctrl+Shift+U
Open Connection	Ctrl+O
Open P4Admin	Ctrl+Shift+A
Diff Against	Ctrl+Shift+D
Show Revision Graph	Ctrl+Shift+R
Show Timelapse View	Ctrl+Shift+T
Open P4V Help	F1
Show System Info	Ctrl+I

The following table lists the default shortcut keys. 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 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 shortcuts

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, open the **Get Revision** dialog and then drag the submitted changelist, workspace, label or file revision to the text field on the right of the **Specify revision using:** option.
- **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, right-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, right-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 respective 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 161.

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
	⬚	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

Category	Icon	Description
		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
		File open for rename/move ("+" indicates target)
		File open for rename/move ("x" indicates source)
		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
Repo Files		Repo file synced to headBlob and at headCommit
		Repo file at headBlob but not at headCommit
		Repo file not at headBlob and not at headCommit
Changelists	✓	Shelved file in pending changelist
		Pending changelist has files that need resolve
		Pending changelist contains shelved files
	▲	Pending changelist has no open files
	▲	Pending changelist not associated with the current workspace or user, with or without open files

Category	Icon	Description
Folders	📁	(Blue folder) A folder in the Helix server depot
	📁	(Yellow folder) A folder in your client workspace
	📁	A folder that maps to a Graph repo
Depots	🗄	Classic depot
	📡	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 Helix server.
	🗄	Stream depot: A depot where stream files are stored.
	🗄	Spec depot: when enabled by your Helix server administrator, a spec depot stores the history of changes to Helix server specifications, such as changelists.
	🗄	Graph depot: A depot for storing files in a repo 📁, which uses the graph model associated with git instead of the "classic" Perforce file revision model.
Workspaces	🖥	Workspace associated with a classic depot
	🖥	Workspace associated with a stream depot
	🖥	Workspace associated with a graph depot, which might or might not be a hybrid workspace
Swarm	▲S	Pending changelist with shelved files in Swarm review
	▲S	Empty pending changelist with shelved files in Swarm review
	●S	Submitted changelist with Swarm review
	S🕓	Folder History with Swarm review

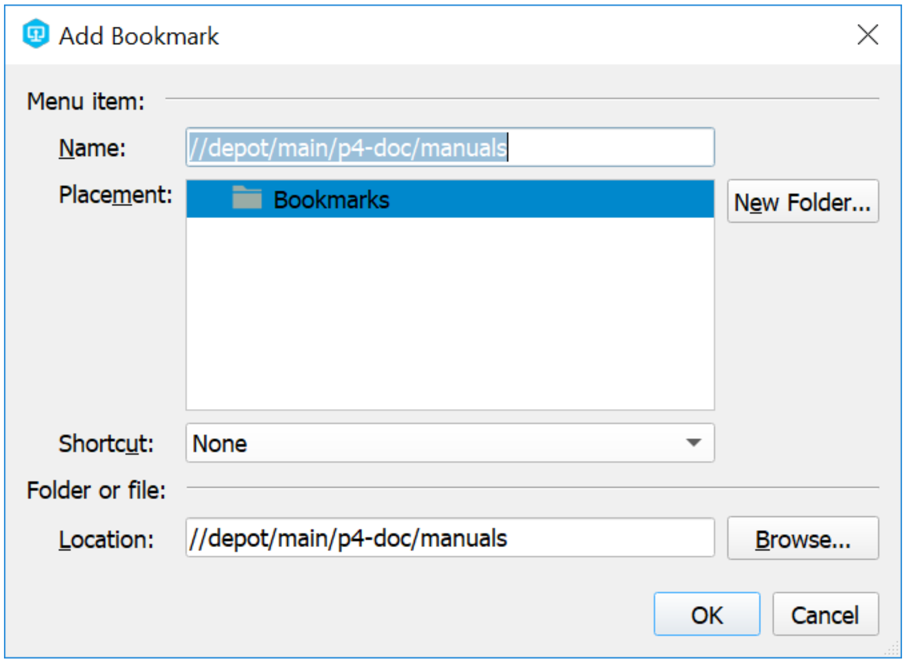
Bookmarking files

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, right-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 **Add Bookmark** dialog is displayed.



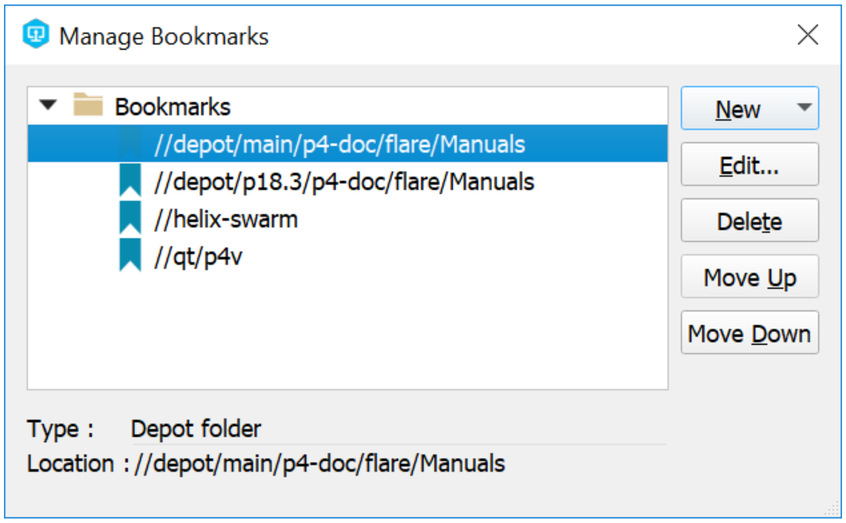
Name: //depot/main/p4-doc/manuals
Placement: Bookmarks
Shortcut: None
Location: //depot/main/p4-doc/manuals

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. In the **Manage Bookmarks** dialog, you can create and edit bookmarks, create folders, and create

separators.



//depot/main/p4-doc/flare/Manuals
//depot/p18.3/p4-doc/flare/Manuals
//helix-swarm
//qt/p4v

Type : Depot folder
Location : //depot/main/p4-doc/flare/Manuals

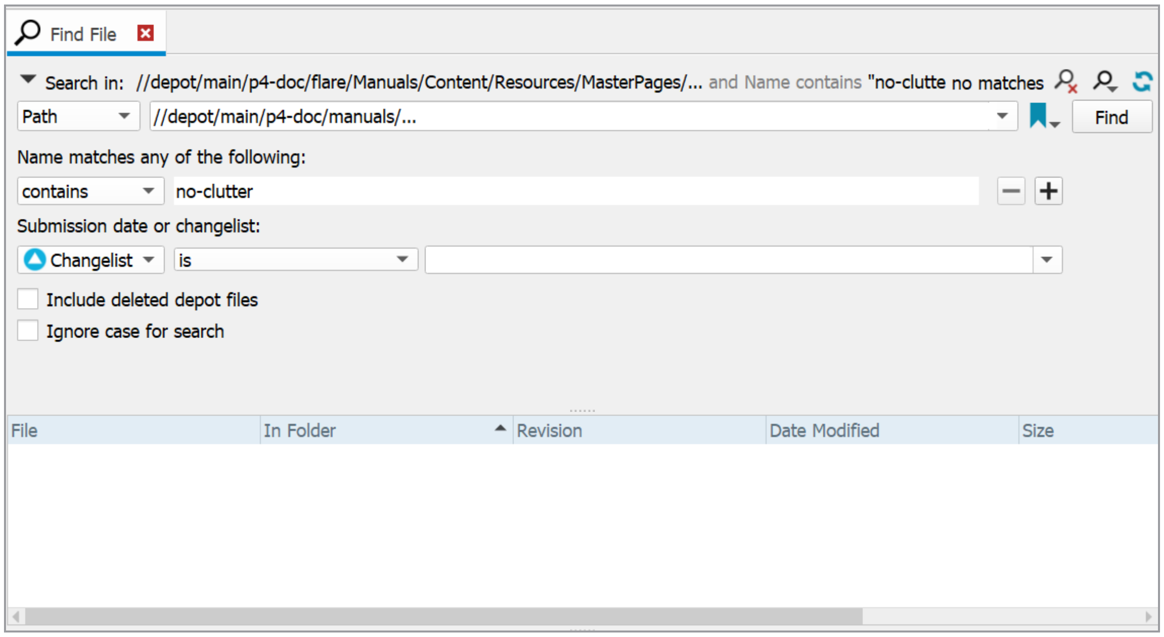
Searching and filtering

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**. The **Find File** tab opens in the right pane.

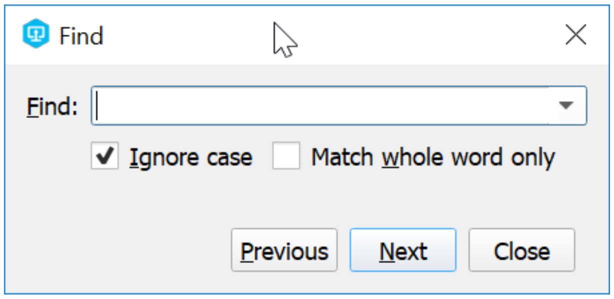

Find File
Search in: //depot/main/p4-doc/flare/Manuals/Content/Resources/MasterPages/... and Name contains "no-clutte no matches
Path
//depot/main/p4-doc/manuals/...
Find
Name matches any of the following:
contains
no-clutter
Submission date or changelist:
Changelist
is
Include deleted depot files
Ignore case for search
File
In Folder
Revision
Date Modified
Size

2. On the **Find File** tab in the right pane, 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**.
   - Select **Ignore case for search**.
4. Click **Find**.

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

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

1. Click **Search > Find** or type +**F** (Windows) or +**F** (Mac). The **Find** dialog opens.



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 specs using filters

You can use filters to find specific labels, submitted or pending changelists, workspaces, and so on. These items are also referred to as 'specifications.'

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

To enter filter criteria for most specification types:

1. Click the Filter disclosure triangle ▶ in the upper pane of the specification tab (Submitted, Jobs, Labels, etc.) to expand the filter pane.
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.

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 84
- For workspaces, "Creating and managing workspaces" on page 23
- For branch maps, "Managing branch mapping" on page 118

- For streams, "Using the stream graph" on page 161
- For jobs, "Managing jobs" on page 123
- For labels, "Managing labels" on page 120

Customize Depot and Workspace views using filters

You can apply filters to your Depot and Workspace views to hide or show certain files and folders.

To customize a tree view:

1. Do one of the following:
   - Click **Search > Filter Depot** or **Filter Workspace**.
   - On the **Depot** or **Workspace** tab, click the filter icon .

2. Select one of the following options:

	Option	Description
Depot tree	**Show Deleted Depot Files**	Select to display files that have been deleted from the depot at head revision.
	Hide Deleted Depot Files	Select to hide files that have been deleted from the depot at head revision.
	Tree Restricted to Workspace View	Select to display only folders and files defined in the workspace.
	Entire Depot Tree	Select to display the whole depot tree.
	Tree Restricted to Depot Type (<*selected value*>)	Click to open the list of depot types, and then select the depot type to display (**Local**, **Stream**, **Spec**, or **Remote**). By default, P4V shows the whole depot tree (**Entire Depot Tree**).
	Tree Restricted to Stream Type	Only available if the depot type selected is **Entire Depot Tree** or **Stream**. Click to open the list of stream types, and then select the stream type to display (**Mainline**, **Development**, **Release**, or **Task**). To clear the selection, select **No Filter**. Note that this option also displays directories that are no longer associated with a stream spec. To show all stream types and hide directories that are no longer associated with a stream spec, select **All Types**.
	No Folder Filter	Select to clear a folder filter (only applies if you have a bookmark selected).
	<path to bookmarked folder>	Select to display only the bookmarked folder and its subfolders in the depot tree.

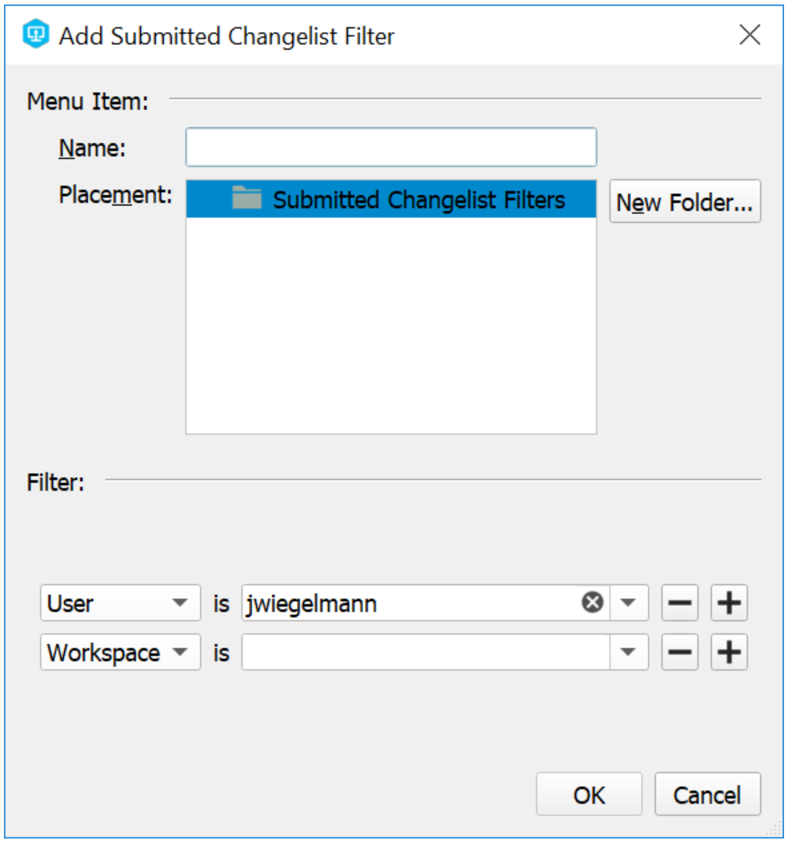
	Option	Description
Workspace tree	**Show Files Not in Depot**	Select to display files that only exist in the workspace, not in the depot.
	Hide Files Not in Depot	Select to not display files that do not exist in the depot.
	Show Only Files Not in Depot	Select to display only files that do not exist in the depot.
	Show Hidden Files	Select to display any hidden files.
	Do Not Show Hidden Files	Select to keep hidden files from displaying.
	Entire Computer	Select to display the entire computer folder hierarchy.
	Workspace Root	Select to display only the workspace root.

The tree view refreshes to reflect your selection.

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**.



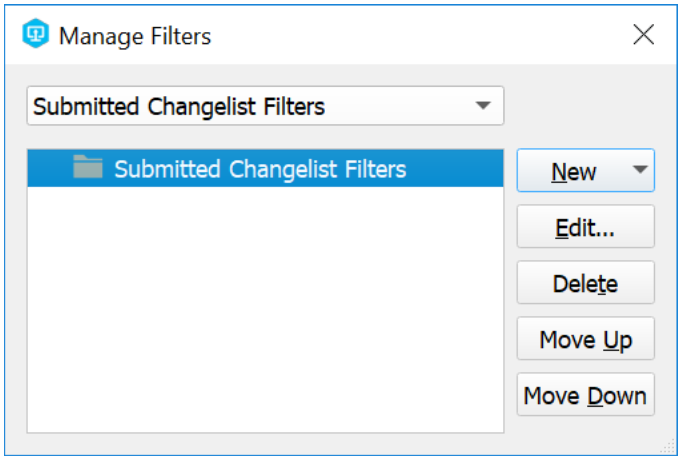
2. In the **Add <*spec type*> Filter** dialog, enter a name and folder location for your saved filter.
3. Click **OK** to save.

To apply a saved filter:

1. In the filter pane, click the **Apply saved filters** icon or go to th e **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. In 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:

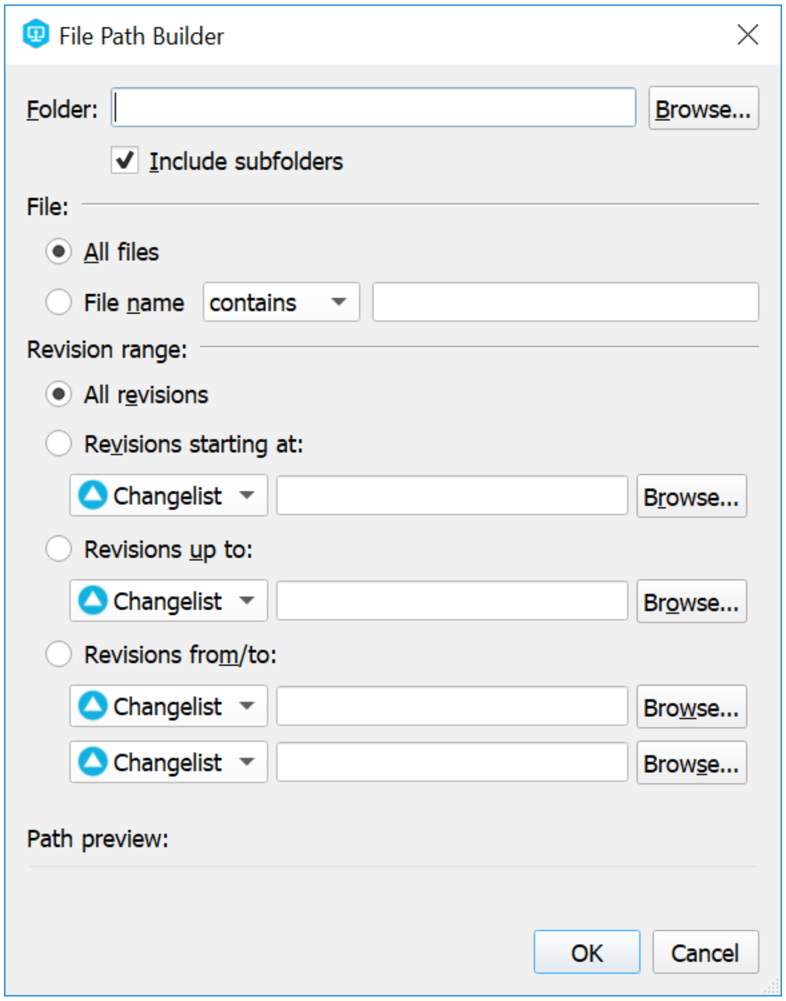
- Use standard Helix server file path syntax (`//depot/folder/folder/filename` or `//depot/folder/...`).

  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 Core Server 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 a 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**, **is**, **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.

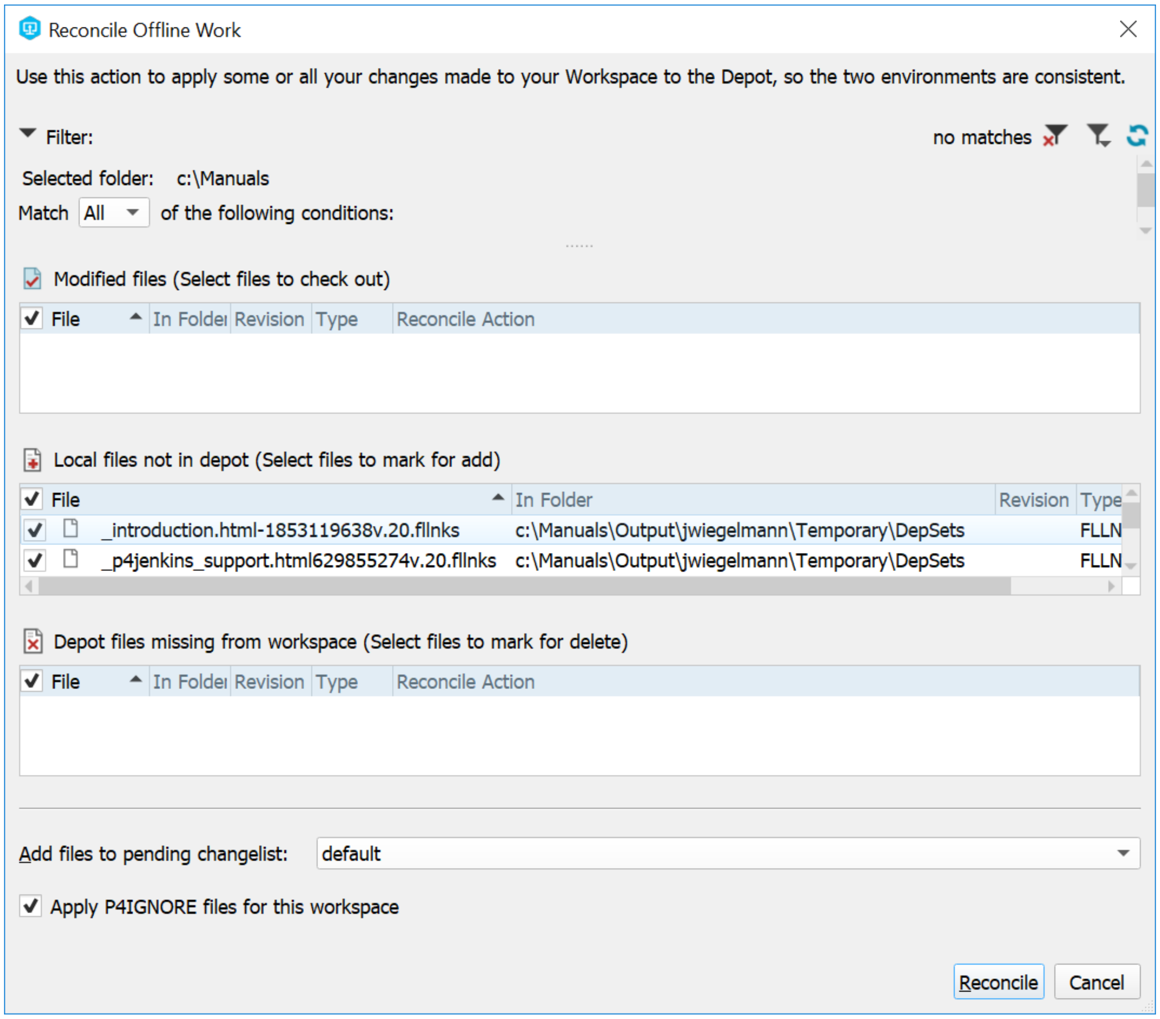
Reconciling offline work

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.

Note
To speed up reconcile operations, you can set the **Modtime** option for the workspace. For details, see "Create a workspace" on page 24.

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

1. In the tree pane, on the **Workspace** tab, right-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.


Reconcile Offline Work

Use this action to apply some or all your changes made to your Workspace to the Depot, so the two environments are consistent.

Filter: no matches

Selected folder: c:\Manuals

Match All of the following conditions:

Modified files (Select files to check out)

File	In Folder	Revision	Type	Reconcile Action

Local files not in depot (Select files to mark for add)

	File	In Folder	Revision	Type
✓	_introduction.html-1853119638v.20.fllnks	c:\Manuals\Output\jwiegelmann\Temporary\DepSets		FLLN
✓	_p4jenkins_support.html629855274v.20.fllnks	c:\Manuals\Output\jwiegelmann\Temporary\DepSets		FLLN

Depot files missing from workspace (Select files to mark for delete)

File	In Folder	Revision	Type	Reconcile Action

Add files to pending changelist: default

✓ Apply P4IGNORE files for this workspace

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 91 and "Define a workspace view" on page 28.

3. In the **Reconcile Offline Work** dialog, do the following:

a. (Optional) Use the filter pane to limit the list of files displayed:

   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*.

   Use the following buttons to add or delete filter rows:

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

b. 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.

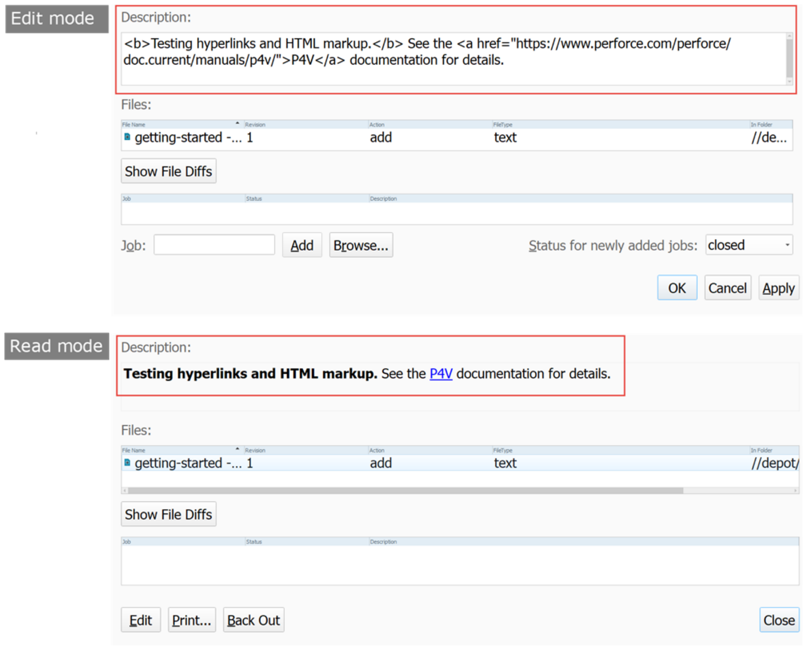
To export files:

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

Formatting text in Description fields

In *edit* mode, P4V supports internal and external drag-and-drop, cut and paste, and undo/redo operations in the **Description** fields of what is generally referred to as specs, such as job specs, workspace specs, stream specs, branch specs, and so on. In addition, you can use HTML syntax to mark up text and create hyperlinks.

In *read* mode, the content of **Description** fields renders in rich text; hyperlinks are active and, when clicked, open up in the default web browser. The following figure illustrates how the content of the **Description** field renders in edit versus read mode.



Edit mode

Description:

```
<b>Testing hyperlinks and HTML markup.</b> See the <a href="https://www.perforce.com/perforce/
doc.current/manuals/p4v/">P4V</a> documentation for details.
```

Files:

getting-started -...	1	add	text	//de...

Job:

Status for newly added jobs: closed

Read mode

Description:

Testing hyperlinks and HTML markup. See the P4V documentation for details.

Files:

getting-started -...	1	add	text	//depot/

Note
For submitted changelists, edit mode is only available if you have a workspace and if the changelist is yours.

The supported HTML tags in **Description** fields are governed by Qt's rich text engine. For details, see Qt's documentation on the supported HTML subset.

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 77

Users with *admin* privileges can also set 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 Core Server Administrator Guide*.

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 191. For more details on setting up the integration, done by an *admin* user on the Helix server, see Swarm integration properties in the *Helix Core Server Administrator Guide*.

To view settings currently in effect, see "Viewing effective settings" on page 79.

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 preferences 59
Streams preferences 59
Server Data preferences 61
Behavior preferences 64
Double-click preferences 70
Shortcuts preferences 70
Logging preferences 70
Display preferences 71
Files and History preferences 72
Fonts preferences 72
Features preferences 73
Tools preferences 73
File Editors preferences 74
Diff preferences 74

Merge preferences 75
HTML Tools preferences 76

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 Helix server. 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 Helix server, so if you switch your connection to a different Helix server instance during a P4V session, you may see different behaviors after you open the new connection. For example, if Helix server A has enabled the **Labels** tab and Helix server B has disabled it, the **Labels** tab will become unavailable when you switch your connection from instance A to instance B.

Note also that there are *performance-related* preferences (such as those on the **Server Data** page) that you can set centrally using Helix server. 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 preferences

You can configure the following settings for connecting to Helix server:

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 Helix server 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.

Whether to **Automatically check for Helix P4V updates**. (See "Checking for updates" on page 19.)

Streams preferences

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 the `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 need 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.
- **Show an information dialog when switching workspaces**: Specifies whether P4V switches workspaces silently when you switch streams.
- **Update files when reusing workspace:** Select one of the following options:
  - **Prompt to update workspace files** to have P4V ask for confirmation when you reuse a workspace with a different stream.
  - **Update workspace to match all files in stream** to have P4V overwrite workspace files without waiting for confirmation.
  - **Do not update workspace files** to keep workspace files unchanged when you switch to a different stream.
- **Switch behavior:**
  - **Use standard switch behavior ("p4 switch")**
    - **Run reconcile before reusing workspace (Server 19.1 or later):** Select to have P4V open the **Reconcile Offline Work** dialog, if applicable, when switching streams in the current workspace. P4V adds the files you choose to reconcile to the default changelist and backs up that changelist when switching streams. This option is only supported with Helix server 2019.1 or later. We recommend that you only use this option when working in a small workspace. In a big workspace, or a workspace with large binary

files, the reconcile operation could take a considerable amount of time. In this case, we recommend that you perform a manual reconcile operation before you switch streams and only on the directory of your choice. For more information, see "Reconciling offline work" on page 54.

- **Use legacy switch behavior (allows switching without having to shelve or revert files that are checked out) - Not recommended**: Select this option if you do not want the p4 switch functionality that helps users shelve or revert checked-out files.

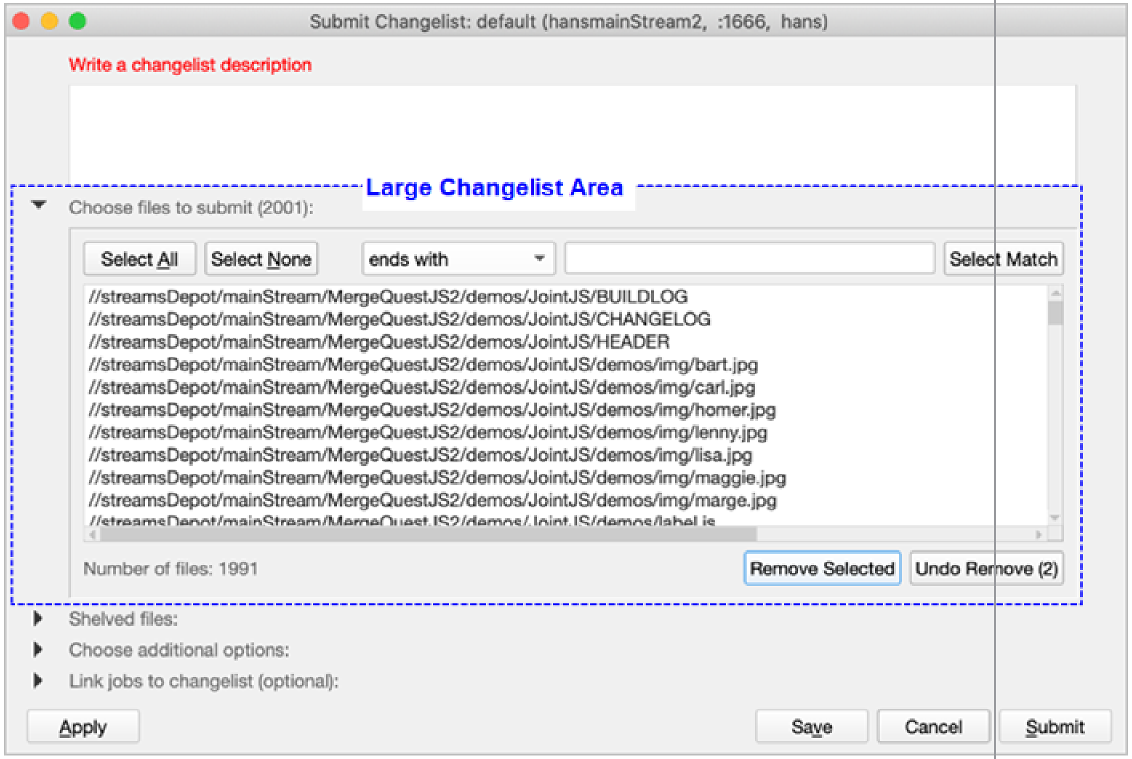
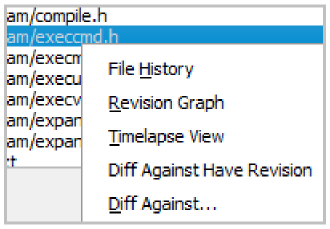
You can control the display properties of the stream graph with the following:

- **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 preferences

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 Helix server for updated file information. Frequent checks enable P4V to display current file status but increase the workload on Helix server.

Maximum number of files displayed per changelis t	Specifies the maximum number of files displayed in a changelist. This setting affects only the display of changelists and does not limit the number of files that a changelist can contain.
	
	The **Submit**, **Shelve**, **Unshelve**, **Revert**, and **Resolve** dialogs display files beyond the maximum number in the Large Changelist Area that allows you to filter the list of files, remove selected files, and provides multiple undo of the removal of files from the list:
	
	
	 Submit Changelist: default (hansmainStream2, :1666, hans)
	Write a changelist description
	Large Changelist Area
	Choose files to submit (2001):
	Select All
	Select None
	ends with
	Select Match
	Number of files: 1991
	Remove Selected
	Undo Remove (2)
	Shelved files:
	Choose additional options:
	Link jobs to changelist (optional):
	Apply
	Save
	Cancel
	Submit
	//streamsDepot/mainStream/MergeQuestJS2/demos/JointJS/BUILDLOG
	//streamsDepot/mainStream/MergeQuestJS2/demos/JointJS/CHANGELOG
	//streamsDepot/mainStream/MergeQuestJS2/demos/JointJS/HEADER
	//streamsDepot/mainStream/MergeQuestJS2/demos/JointJS/demos/img/bart.jpg
	//streamsDepot/mainStream/MergeQuestJS2/demos/JointJS/demos/img/carl.jpg
	//streamsDepot/mainStream/MergeQuestJS2/demos/JointJS/demos/img/homer.jpg
	//streamsDepot/mainStream/MergeQuestJS2/demos/JointJS/demos/img/lenny.jpg
	//streamsDepot/mainStream/MergeQuestJS2/demos/JointJS/demos/img/lisa.jpg
	//streamsDepot/mainStream/MergeQuestJS2/demos/JointJS/demos/img/maggie.jpg
	//streamsDepot/mainStream/MergeQuestJS2/demos/JointJS/demos/img/marge.jpg
	//streamsDepot/mainStream/MergeQuestJS2/demos/JointJS/demos/label.js
	
	This area displays the number of files in the changelist and reflects how many files you removed from the changelist.
	
	To see file history, right-click any file in the list:
	
	
	 am/compile.h
	am/execcmd.h
	am/execm
	am/execu
	am/execv
	am/expan
	am/expan
	File History
	Revision Graph
	Timelapse View
	Diff Against Have Revision
	Diff Against...

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 Helix server 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`
Maximum number of files to display in Dashboard Workspace Folder View	Limits the number of files displayed in the **Dashboard** view.
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 Core Server Administrator Guide*.

Disable parallel submit	Disables concurrent transfer of files when submitting to a 2015.1 or later 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. For information on the server configurables pertaining to parallel submit, see `net.parallel.submit.threads` and `net.parallel.max` in the *Helix Core P4 Command Reference*. Parallel transfer mode only kicks in if you set **`net.parallel.max`** to a value larger than `0`
Disable parallel shelve	Disables file transfer in parallel mode. Parallel shelving is enabled when the **`net.parallel.shelve.threads`** config parameter is set in a server (2017.2 or later). By default, this setting is turned on. For more information on the server configurable, see `net.parallel.shelve.threads` in the *Helix Core P4 Command Reference*.
Automatic Safe Resolve (no merging) when syncing files	Automatically resolves differences when getting a file revision by either accepting the target file or the source file, depending on which file has changes. By default, this setting is turned on. When turned off, P4V opens the **Resolve** dialog instead to let you decide how to resolve differences.

Behavior preferences

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.
- **Update files when modifying workspace mappings:** Select one of the following options:
  - **Prompt to update workspace files** to have P4V ask for confirmation when modified workspace mappings affect the files in your workspace.
  - **Automatically update workspace** to have P4V overwrite workspace files without asking for confirmation.
  - **Never update workspace files** to keep workspace files unchanged when you modify workspace mappings.
- **Prompt for name when creating new workspace:** Specifies whether P4V displays the **Workspace Name** dialog when you create a new workspace. By default, this prompt is turned off.

  Note
  If this feature is turned on on the server side, selecting or clearing this check box has no effect. You cannot override server-side configurations.

- **Warn before checking out files, if the number of files exceeds <*xxx*>**: Select to receive a warning when checking out a large number of files. The default limit is 1000 files.

  Note
  Selecting this option may cause performance issues.

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.
- When dropping file(s):
  - **on a file, do a diff comparison:** Launches the **Diff** dialog when you drop a file on another file.
  - **anywhere within a changelist, move open files to new changelist:**
  - **on a file, do nothing:**

Merge-Integrate preferences

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 preferences

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.
- **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.

Warning
The following integration flags can have unexpected or undesired results. Do not select them if you are not certain you want these actions to be applied.

- **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 (-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).
- **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 Core server 2011.1 and up	`p4 copy`	`-v`

	Condition	Command	Available Flags
Branch	Helix Core server 2010.2 and earlier	`p4 integrate`	`-v`
Branch	Helix Core server 2011.1 and up	`p4 populate`	None

For more information about these P4 commands, see the *Helix Core P4 Command Reference*.

Copy preferences

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 preferences

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

  Note
  The branch method is always:
  - *Specify source and target files* for a stream object
  - *Branch mapping* for a branch mapping

- Specify how to treat the files that you use to filter a branch mapping:
  - **Source:** Files that you include are considered the source
  - **Target:** Files that you include are considered the target
  - **Remember my last choice:** Repeat the previous use of the dialog for Source or Target
- 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.

To return to the original settings of P4V, click **Restore Defaults**.

For more information about these options, see Creating Branches.

Double-click preferences

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 preferences

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 preferences

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 Helix server.
- **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 preferences

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 Helix server (**Server time**) or for your local client.
- **Scale application icons based on (requires restart):** Select **Calculated primary monitor resolution** if you mainly view P4V on your primary monitor. If you view P4V on a secondary monitor with a different resolution than the primary monitor, select **Custom size** and move the slider to the scaling that is right for your secondary monitor.
- **Application color scheme (requires restart):** Select **Light theme** (default) for the classic P4V colors. Select **Dark theme** if you prefer light font on a dark background.

Localization:

- **Language used for application menus, labels, and dialogs (requires restart):** Select English or one of the installed languages.
- **Set encoding for all connections to:** Sets the character encoding for connections to unicode-mode Helix server.

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 Helix server. The encoding that you set here does not affect 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 preferences

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 (on by default).
- **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.
- **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
- **Show changelists integrated into the specified folder (-i):** Includes integrations with folder history.

Fonts preferences

You can configure the fonts to be used for the P4V UI and for file contents displayed in P4V.

To configure fonts:

1. Go to **Edit > Preferences**.
2. In the **Preferences** dialog, on the **Fonts** tab, do the following:
   - To configure the font to be used in the P4V UI, under **Application font**, select the font family, font style, and font size from the respective drop-down lists.
   - To configure the font to be used when displaying file contents, under **File content font**, select the font family, font style, and font size from the respective drop-down lists.

     If you prefer a monospace font type, select **Show fixed sized fonts only** to hide font types not matching this criterion.

   The **Sample** area displays a preview of your selections.
3. Click **OK** to save your changes and to close the dialog.

If you made changes to the file content font, you should see the new font take effect when you re-open the **History** tab. The **Preview** tab in the **Details** pane (located at the bottom of the **History** tab) now displays the content of files in text format in the newly configured font.

For changes to the application font to take effect, continue with step 4.

4. Restart P4V for your changes to take effect.

Features preferences

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 preferences

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

Revision Graph:

- **Limit Revision Graph to ancestors and descendants**: Limits a file’s integration history to ancestors and descendants (default). This option has the smallest footprint and ensures optimized performance.

- **Show Full Revision history in Revision Graph:** Displays the full integration history of the branch. With this option, the revision graph might take longer to display.

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 preferences" below.

Image Timelapse preferences

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 preferences

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 right-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 preferences

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 preferences

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.

HTML Tools preferences

HTML tools in P4V are tied to the feature property P4V`.Features.HTMLTools` (for details, see the topic Feature-related P4V properties in the *Helix Core Server Administrator Guide*). When this property is turned on (which is the default), P4V can run in the following modes:

- **No HTML Tools:** In this mode, neither HTML tools nor applets are enabled. You cannot extend P4V.
- **Enable HTML Tools:** In this mode, P4V supports P4VJS and provides the editors to add custom HTML windows and tabs. When P4VJS demos are available in the P4V Resources directory, you can run P4V in P4VJS demo mode. For more information, see the *P4VJS User Guide*.
- **Allow Applets:** In this mode, P4V supports the legacy P4JsApi.

Note
For custom HTML pages that you add using HTML tools, P4V provides additional security by letting you specify a whitelist of base URLs. For example, if the following base URLs are white-listed, P4V only accepts definitions and pages that match these base URLs:

- http://www.toolserver.com/customtabs
- http://www.toolserver.com/visualtools
- http://www.toolserver.com/pages
- http://www.toolserver.com/tabs

A URL that does not start with the base path listed in these URLs fails with a security exception error. Local files are exempt.

To enable HTML tools in P4V:

1. Go to **Edit > Preferences** (Windows) or **P4V > Preferences** (Mac).
2. In the **Preferences** dialog, on the **HTML Tools** tab, select **Enable HTML Tools**.

3. If you want to restrict the locations from which HTML tools can run:
    a. Select the **Only accept HTML Tools from** check box.
    b. To the right of the **Permitted BaseURLs** filed, click **Add**.
    c. In the **Accepted BaseURL** dialog, in the **URL** field, specify a URL to whitelist it and click **OK**.
    d. Repeat steps b and c as needed.
4. Click **OK**.
5. Restart P4V.

> **Note**
> HTML Tools support cookies. If you want to delete cookies, use **Preferences > HTML Tools > Advanced Settings > Delete all cookies**.

Editing user preferences

In addition to your name and email address, user preferences in P4V include what jobs, if any, you automatically want to attach to new changelists that you submit (such as all open jobs assigned to you) and what files or folders you want to review. When a file or folder is under review, it means that you keep a watch on it and get notified if changes occur in a specific file or any of the files in a folder.

To configure user preferences:

1. Go to **Connection > Edit Current User**. The **User** form opens.
2. On the **Form** tab, configure the following:
    - **Full Name:** Enter your full name as you want it to appear to other users.
    - **Email:** Enter the email address that you want to use 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 changelist 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 "Search for jobs" on page 124.
    - **Reviews:** Enter the files that you want to watch. You will receive an 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
```

You receive 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.

3. If you prefer to select depot paths instead of typing them, on the **Reviews** tab, do the following:
   a. If needed, expand the tree to navigate to a folder.
   b. Do one of the following:
      - Right-click the folder name to view a menu of inclusion choices:
        - **Include** or **Exclude Tree:** Select to include or exclude all files in all subfolders within the selected folder.
        - **Include**or **Exclude Files:** Select to include or exclude all files in the folder but none within subfolders.
        - **Include** or **Exclude Special…:** Select to 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.
      - Double-click the folder name to open the **Special Edit of View Map** dialog, where you can specify filenames, extensions, or expressions written in standard Helix server filter expression syntax.
   c. Click **OK**.

To change your password:

1. Go to **Connection > Change Password**.
2. In the **Change Password** form, enter your old password and your new password.

   The password must include:
   - At least 8 characters
   - Upper- and lowercase letters, or letters plus one or more symbol or number
3. Confirm you new password.
4. Click **OK** to confirm your changes.

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 **80**
Retrieving files from the depot **81**
Editing files **82**
Reverting files **82**
Checking in files **84**
View changelists 84
Submit changelists 84
Reverse a changelist submission 85
Restrict access to a changelist 86
Configure changelist display 86
Undoing changes **87**
Displaying revision history **88**
Files 89
Folders 89
Changelists 90
Changing a file's type **90**
Renaming and moving files or folders **91**
Cleaning up files and directories **91**
Deleting files **92**
Diffing files and folders **92**
Diff dialog options 93
Check workspace consistency 94
View the state of the depot at a specified point in time 94
Diff large files 94
Shelving files **95**
Shelve checked-out files in a pending changelist 96
Unshelve files 96
Submit shelved files 97
Delete shelved files 98

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. Right-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, right-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 Files Not In Depot**.

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 41 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, right-click the file and choose **Get Latest Revision**.

To get a previous revision, right-click the file and choose **File History**. The **History** tab opens. This tab displays the revision history of the file or folder selected in the tree pane. In the **History** tab, right-click the desired revision and choose **Get this Revision**. Alternately, right-click the file and choose **Get Revision**. In the **Get Revision** dialog, specify the desired revision and click **Get Revision**.

To get the revisions or the previous revisions for files in a changelist, open the **Submitted** tab. This tab displays a list of submitted changelists, based on the filter criteria selected at the top of the tab. Its content is not related to any files or folders selected in the tree pane. In the **Submitted** tab, right-click the desired changelist and choose **Get Revisions For Files in Changelist <*xxx*>** or **Get Previous Revisions for Files in Changelist <*xxx*>**.

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

Tip
To get a revision into your workspace even if the workspace already has the file, in the **Get Revision** dialog, select **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. Right-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 right-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, right-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.
3. If P4V detects any changes, the **Revert** dialog box opens:

   a. Select the changes you want to revert.

   b. If you also want to delete files that were open for add from the workspace, select the relevant check box.

      Otherwise, P4V removes the add request but keeps the file in the workspace.

   c. To prevent the Revert dialog from opening when reverting files, select the relevant check box.

   d. Click **Revert**.

   e. If you chose to delete files that were open for add from the workspace, click **Delete Files** to confirm.

   > **Warning**
   > You cannot revert this action.

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 Helix server. In addition, a default changelist is maintained for each client workspace. If submission of the default changelist fails, Helix server 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 right-clicking the file name in the Tree pane or by selecting a pending changelist from the **Pending** tab.

If a pending changelist includes shelved files, you first need to unshelve or delete those files, either prior to trying to submit the changelist or from the **Submit** dialog.

To check in individual files:

1. Right-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.

   The **Description** field accepts HTML tags for marking up and hyperlinking text. For details, see "Formatting text in Description fields" on page 57.
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. If the changelist includes any shelved files, you need to delete or unshelve them before you can submit the changelist. Do any of the following:
   - Delete shelved files: Above the **Shelved files** area, click **Delete selected**.
   - Unshelve shelved files: In the **Shelved files** area, right-click the files and select **Unshelve**.

     In the **Unshelve** dialog, edit as needed and click **Unshelve**.

   Alternatively, you can submit the shelved files directly prior to submitting the changelist.
5. Click **Submit**.

To edit the description of a submitted changelist:

1. Right-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 all files from one pending changelist to another:

Right-click the changelist and select **Move All Files to Another Changelist**.

To move a file from one pending changelist to another:

1. Expand the source changelist.
2. Do one of the following:
   - Drag the file to the target changelist.
   - Right-click the file or files and select **Move to Another Changelist**.

Reverse a changelist submission

You can restore the state of a file or folder as follows:

- Back out a single change or a range of changes by specifying a changelist, revision number, date/time, or label, or a range of the same, and keep subsequent changes.
- Roll back to the state of a given changelist, date/time, or revision number.

For details, see "Undoing changes" on page 87.

Note
With Helix server 2016.2 and earlier, undoing changes works as described in Reverse a changelist submission in the *P4V User Guide* for version 2017.2.

Restrict access to a changelist

To restrict who can see a changelist, select the **Restrict Access to Changelist** option when editing a pending or submitted 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.

The visibility of restricted changelists:

- **Pending changelists**: Visible only to owner, regardless of whether other users have access to checked-out files.
- **Pending changelists containing shelved files**: Users with *list* (or higher) permission (as specified in the protection table) to one or more of the shelved files can view those files but cannot view the changelist description.
- **Submitted changelists**: Users with *list* (or higher) permission (as specified in the protection table) to one or more of the submitted files can list those files and read the changelist description.

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

With Helix server 2017.1 and later, you can undo:

- A single change made at a given changelist or revision while keeping subsequent changes.
- A range of changes made at a given range of changelists or dates, or at a given label (of which you are the owner), while keeping subsequent changes.
- All changes from a selected changelist, date, or label to the most recent version. The submitted files become the new head revision in the depot.

Note
Helix server does not undo integration records with `p4 undo`. When P4V undoes a merge event, it does undo the change but does not undo any integration history.

The **Undo Changes** dialog always opens in the context of the selected entity, which can be a workspace or depot folder or file, a changelist, a revision, or a label. For example:

- If you open the dialog from a changelist, you can browse to a subfolder to narrow down the operation to files in a subfolder. The folder you select must match the selected changelist.
- If you open the dialog from a revision, it opens in the context of the selected revision. You can specify whether to undo only the specified revision, the specified revision up to another revision, while keeping subsequent changes, or all changes from the selected revision to the head revision.
- If you open the dialog from a folder, you need to specify whether to base the operation on a specific revision or changelist or, if you want to undo a range of changes, on a changelist, date/time, or a label.

When undoing a range, the **Undo** option creates the range according to the limit type:

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

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.

Important
When undoing rename or move operations, make sure to perform the undo option on the changelist that includes the rename or move operation. Otherwise, P4V will display an error message. `p4 move` and `p4 rename` operations are atomic in that you cannot split the move/add and move/delete pair.

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.

Note
With Helix server 2016.2 and earlier, undoing changes works as described in Reverse a changelist submission in the *P4V User Guide* for version 2017.2.

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 submitted changelist on the **Submitted** tab
2. From the **Actions** menu, select **Undo Changes**.
3. In the **Undo Changes** dialog, if needed, specify whether you want to undo a single change, a range of changes, or all changes from a selected point forward.
4. Depending on your selection in step 3, specify a revision, changelist, date/time, or label, or a range of the same.
5. Specify whether to check out the changes to a new or existing changelist.
6. Optionally, to inspect the file affected by the changes, click **Preview**.
7. 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.

Displaying revision history

This section explains how to view the revision history of a file, folder, or changelist. For information on file revisions in labels, see "Retrieve file revisions in a label" on page 123

P4V displays a file's revision history, including integration and label history, on the **History** tab. This tab works in conjunction with the **Depot** or **Workspace** tree. P4V gets all file revisions for the file path or directory selected in the tree, up to the selected changelist. If a file revision is currently in your workspace, it is indicated with a red box: .

Files

To display a file’s revision history:

1. Select a file in the **Depot** or **Workspace** tree.
2. Do one of the following:
   - From the **View** menu, select **History**.
   - Right-click and choose **File History**.

   The **History** tab opens, listing the revision history for the selected file.

From here, you can:

- **Get a selected revision into your workspace:** Right-click a *revision* on the **History** tab or a file inside a changelist on the **Submitted** tab and choose **Get This Revision.**
- **Get a different revision into your workspace:** Right-click a *file* on the **History** tab, in the **Depot** tree, or in the **Workspace** tree, and choose **Get Revision**. In the **Get Revision** dialog, specify the revision you want to get, any applicable options, and click **Get Revision**.
- **View a preview of a revision’s content:** Select the revision on the **History** tab; then 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.
- **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.
- **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:

1. Select a folder in the **Depot** or **Workspace** tree.
2. Do one of the following:
   - From the **View** menu, select **History**.
   - Right-click and choose **File History**.

   The **History** tab opens, listing the revision history for the selected folder.

From here, you can:

- **Compare two folder revisions:** Click and drag one revision to the other. P4V launches the Folder Diff Utility.

Changelists

You can get revisions of a changelist from the **Submitted** tab. Note that this works independently from the **Depot** or **Workspace** tree. In the **Submitted** tab, P4V gets the revision of all the files in the changelist at the specified changelist, regardless of which file or folder is selected in the tree on the left.

To display a changelist's revision history:

1. From the **View** menu, select **Submitted Changelists**.

   P4V opens the **Submitted** tab, listing submitted changelists.

2. Do one of the following:

   - To get the revision of all files in a changelist: Right-click the changelist and choose **Get Revisions for Files in Changelist <*xyz*>**.
   - To get the revision of all files in multiple changelists: Right-click the selected changelists and choose **Get Revisions for Files in Changelists**.

   Warning
   These options immediately force-syncs the files of the respective changelist or changelists to your workspace client. There is no intermediary step that provides you the option to cancel the operation.

   Alternatively, to get a different revision of a changelist into your workspace, right-click the changelist and choose **Get Revision**. In the **Get Revision** dialog, specify the files you want to get, the revision number, any applicable options, and click **Get Revision**.

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. Right-click the desired file and choose **Check Out**.

   The file is checked out.

2. Right-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" the in the *Helix Core 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. Right-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. Right-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. Right-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, right-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. Right-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. Right-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 right-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. Right-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 right-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. Right-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 54.

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%\p4enviro.txt
     ```

   - On Linux platforms:

     ```
     export P4ENVIRO=~/.p4enviro
     ```

2. Depending on your platform, create the file `%USERPROFILE%\p4enviro.txt` or `~/.p4enviro` 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 *Helix Core 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 *un*shelve 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. Right-click the changelist and select **Shelve**.

   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.

   P4V displays the **Shelve** dialog. By default, all files in the changelist are selected.
2. Clear the check boxes of any files that you do *not* want to shelve.
3. Select any of the following options that apply:
   - **Revert checked out files after they are shelved:** Reverts the files in your workspace to the head revision in the depot. By default, when this option is selected, P4V also removes any files that are marked for add in the changelist from the file system when they are shelved. If you do not want P4V to remove these files from the file system when they are shelved, make sure to clear the **Remove files that are opened for add** check box.
   - **Don't shelve unchanged files:** Only shelves files that have changed.
   - **Make shelf globally accessible:** Promotes shelved files from an edge server to a commit server, where it can be accessed by other edge servers participating in the distributed configuration. Once a shelved change has been promoted, all subsequent local modifications to the shelf are also pushed to the commit server and remain until the shelf is deleted.
4. Click **Shelve**.

   P4V shelves the file in the selected changelist (or, if you are shelving files in the default changelist, creates a new changelist).

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 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 files in a pending changelist:

1. Right-click the file changelist and select **Unshelve**.

   P4V displays the **Unshelve** dialog. By default, all files in the changelist are selected.
2. Clear the check boxes of any files that you do *not* want to unshelve.
3. Select any other desired options.

4. Click **Unshelve**.

   The shelved files are copied to your workspace and opened in the specified changelist.

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. In the **Unshelve** dialog, select the **Map unshelved files** check box.
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.
3. Click **Unshelve**.

Submit shelved files

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

Note
If a pending changelist includes non-shelved files or a checked-out stream along with shelved files, you must first revert the non-shelved files or the checked-out stream or move them to another changelist.

You cannot submit shelved files from a task stream.

To submit shelved files in a pending changelist, right-click the changelist or shelved files folder and select **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,** right-click the file and choose **Delete**. Alternately, right-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 99
Read the revision graph 99
Navigate the revision graph 100
Filter the revision graph 101
Display details 101
Viewing file history with Time-lapse View 101
Toolbar 102
Slider 103
Viewing image file history with Time-lapse View 104
Using the folder diff utility 105
Diffing text files 106
Diffing images 107
Merging files 108

Additional codeline management tools are described in "Managing codelines" on page 109.

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, or when a revision of the file was undone. You can view multiple Revision Graphs at the same time.

To launch the Revision Graph for a file:

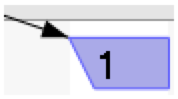
1. Select a file or folder in the **Depot** or **Workspace** tree, or a file or revision in the **History** tab.
2. On the toolbar, click .

   The Revision Graph opens in a separate window.

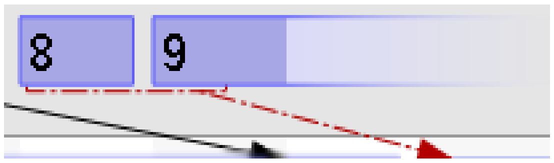
Alternatively, you can also right-click any of these entities and select **Revision Graph**.

Read the revision graph

In 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, the 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 names of the shapes in the legend reflect the action that occurred during the integration and resolve process that created the revision. You can view more details about the actions related to a particular revision by selecting the revision in the graph and then looking at the **Integrations** tab in the lower left pane. For details on the integration actions, see `p4 integrated` in the *Helix Core P4 Command Reference*.

To view details about a revision, such as the changelist number, date, or action performed, click the **Details** tab in the lower left pane. **To view the changelist** (or sync to it or integrate it), click the changelist number in the **Details** tab or right-click a revision in the graph and select **View Changelist**.

Navigate the revision graph

To select revisions, click them or use the arrow keys. Details about the selected revision are displayed in the **Details** tab in the lower left pane. **To select two revisions**, control-click (Windows) or command-click (Mac) them. In this case, the Navigator and Legend tabs disappear and P4V displays a second set of **Details**, **Integrations**, **Labels**, and **Preview** tabs in the lower right pane.

For files that have a large history, the 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** tab outlines the portion displayed in the main window.

To navigate the diagram, do one of the following:

- Drag the box inside the **Navigator** pane.
- Use the scrollbars in the main window pane.
- In the main window pane, 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 right-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.

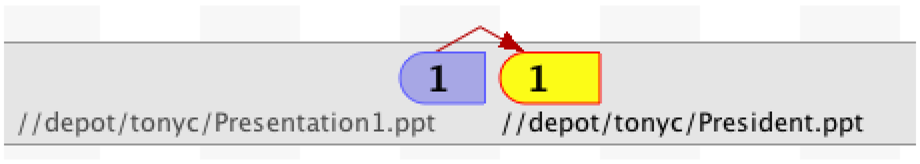
Filter the revision graph

To reduce the detail displayed in the main window pane, you can filter the information:

- **To remove a file or folder from the main window**, clear the respective check box in the **File Filter Tree** on the left.
- **To enter a more precise file filter**, click **Filter Options** and enter the file specification for the files and folders you want to retain in the main window pane (or, for files and folders you want to exclude, exclusionary lines preceded by "-"), check any filtering options you want to apply, and then click **Filter**. To retain this filter in effect for future invocations of the Revision Graph, click **Set As Default**.

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

- **File Renames Collapsed**: Displays the original and renamed files on a single line instead of multiple lines by omitting intervening revisions. An angled arrow indicates the operation, as shown in the following figure.



- **Compressed Integration History**: Displays only revisions that were branched or integrated.

To switch between displaying only ancestors and descendants (linear history), which is the default view, and showing the full revision history, use the branch history button. To change the default behavior, go to **Edit > Preferences > Tools**. For details, see "Tools preferences" on page 73.

Display details

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

Related revisions are listed on the **Integrations** tab. **To get the revision, diff it, or display its history,** right-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 104.

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

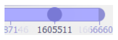
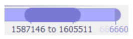
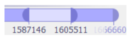
Toolbar

Mode	Determines how many revisions are displayed. Options are: ■ **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 *originating changelist.*
Find	Search text
Go to line number	In single revision mode, go to specified line number.
Go to Next diff	Go to next change.
Go to Previous diff	Go to 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 101.

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. Right-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 2.00	Specifies the time between image revisions during Autoplay.
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.
🟦	Turquoise highlight indicates file or folder has moved.

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 45.
⬅	Go to previous diff
➡	Go to 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

P4V lets you compare text files and image files to navigate their differences. To do a diff on the selected files, P4V launches P4Merge. 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.

By default, P4Merge displays diffs in a side-by-side layout. However, you may prefer to view changes closer to each other. In this case, you can switch to single-pane layout. To display diffs in a single pane, go to **View > Single Pane Diff Layout**. In single-pane mode, deleted text is shown using strike-through text.

To toggle the display of line numbers, click ☰.

For more information, see Navigate diffs in the *P4Merge User Guide*.

Diffing images

P4V lets you diff the following image file types:

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

You can compare two different image files or two revisions of the same image. In both cases, P4V launches P4Merge, where you can navigate the differences between the specified images.

For more information, see View image differences in the *P4Merge User Guide*.

To diff two different image files:

1. In P4V, right-click an image file and choose **Diff Against**.
2. In the **Diff** dialog, specify the other image and click **Diff**.

Alternatively, in the Depot or Workspace tree, or in the **Files** tab, you can drag one image file's icon onto another.

To diff two revisions of the same image file:

1. In P4V, right-click the image file and choose **File History**.
2. In the **History** tab, select the revisions you want to compare, right-click, and select **Diff Selected**.

Alternatively, you can drag one of the desired revisions onto the other revision.

Merging files

If you and another user have edited the same file, P4V 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 is 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.

For more information on resolving changes, including opening P4Merge, see "Resolving files" on page 113.

For more information on merging changes in P4Merge, see Navigate diffs and Merge text in the *P4Merge User Guide*.

6 | Managing codelines

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

Creating branches 109
Merging files between codelines 110
Open files for merge 111
Resolving files 113
Resolve individual files 113
Resolve multiple files 116
Managing branch mapping 118
Work with branch mapping 118
Managing labels 120
Create labels 120
Label files 121
Display and search for labels 121
Edit labels 122
Delete and unload labels 122
Retrieve file revisions in a label 123
Display files associated with a label 123
Managing jobs 123
Create a job 124
Add a job to a pending changelist 124
View jobs 124
Search for jobs 124

For more information about viewing file history, including branching history, see "Displaying revision history" on page 88 and "Viewing codeline history in the revision graph" on page 99.

For more information about using streams, which provide a very structured branching strategy, see "Working with streams" on page 141.

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. If you are hesitating whether to use streams, read the "About streams" on page 141 section before you continue to get a better understanding of the advantages that streams entail.

To branch a codeline or stream:

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

   The **Branch** dialog is displayed.
2. In the **Choose target files/folders** 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
In **Helix server** terminology, the term *branch* refers to a codeline, but it 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 a 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 Core Server 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 right-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 *Helix Core 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 right-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.

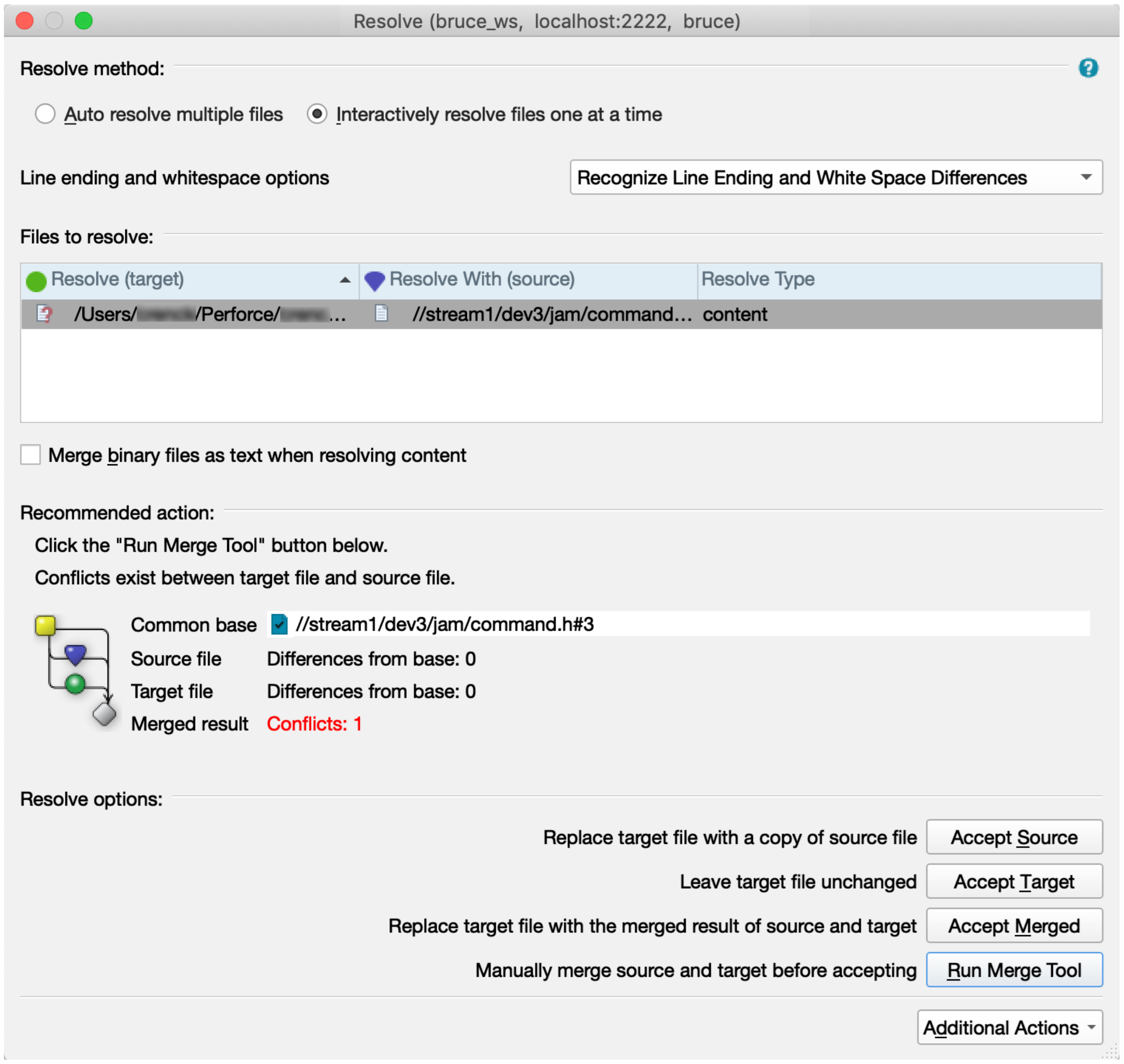
You can resolve conflicts resulting from content changes, filetype changes, attribute changes, moves, deletes, and branching.

Resolve individual files

To resolve an individual file:

1. Right-click the file and select **Resolve**.

   The **Resolve** dialog opens in interactive mode by default.


Resolve (bruce_ws, localhost:2222, bruce)

Resolve method:

Auto resolve multiple files [✓ Interactively resolve files one at a time]

Line ending and whitespace options: Recognize Line Ending and White Space Differences

Files to resolve:

Resolve (target)	Resolve With (source)	Resolve Type
/Users/[redacted]/Perforce/[redacted]...	//stream1/dev3/jam/command...	content

☐ Merge binary files as text when resolving content

Recommended action:

Click the "Run Merge Tool" button below.

Conflicts exist between target file and source file.

Common base	//stream1/dev3/jam/command.h#3
Source file	Differences from base: 0
Target file	Differences from base: 0
Merged result	Conflicts: 1

Resolve options:

Replace target file with a copy of source file	Accept Source
Leave target file unchanged	Accept Target
Replace target file with the merged result of source and target	Accept Merged
Manually merge source and target before accepting	Run Merge Tool

2. 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. By default, P4V recognizes line endings and white space differences. If you want P4V to ignore line endings or white spaces, select an option from the **Line ending and whitespace options** list.

- **Ignore Line Ending Differences:** Ignores differences in line-ending convention
- **Ignore Line Ending and White Space Length Differences:** Ignores whitespace-only changes (for example a tab replaced by eight spaces)
- **Ignore Line Ending and All White Space Differences:** Ignores whitespace altogether (for example deletion of tabs or other whitespace)

If you select any of these options and the files differ by whitespace only, P4V resolves the files using the text in the workspace file.

4. (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.
5. Select whether to **Merge binary files as text when resolving content.**

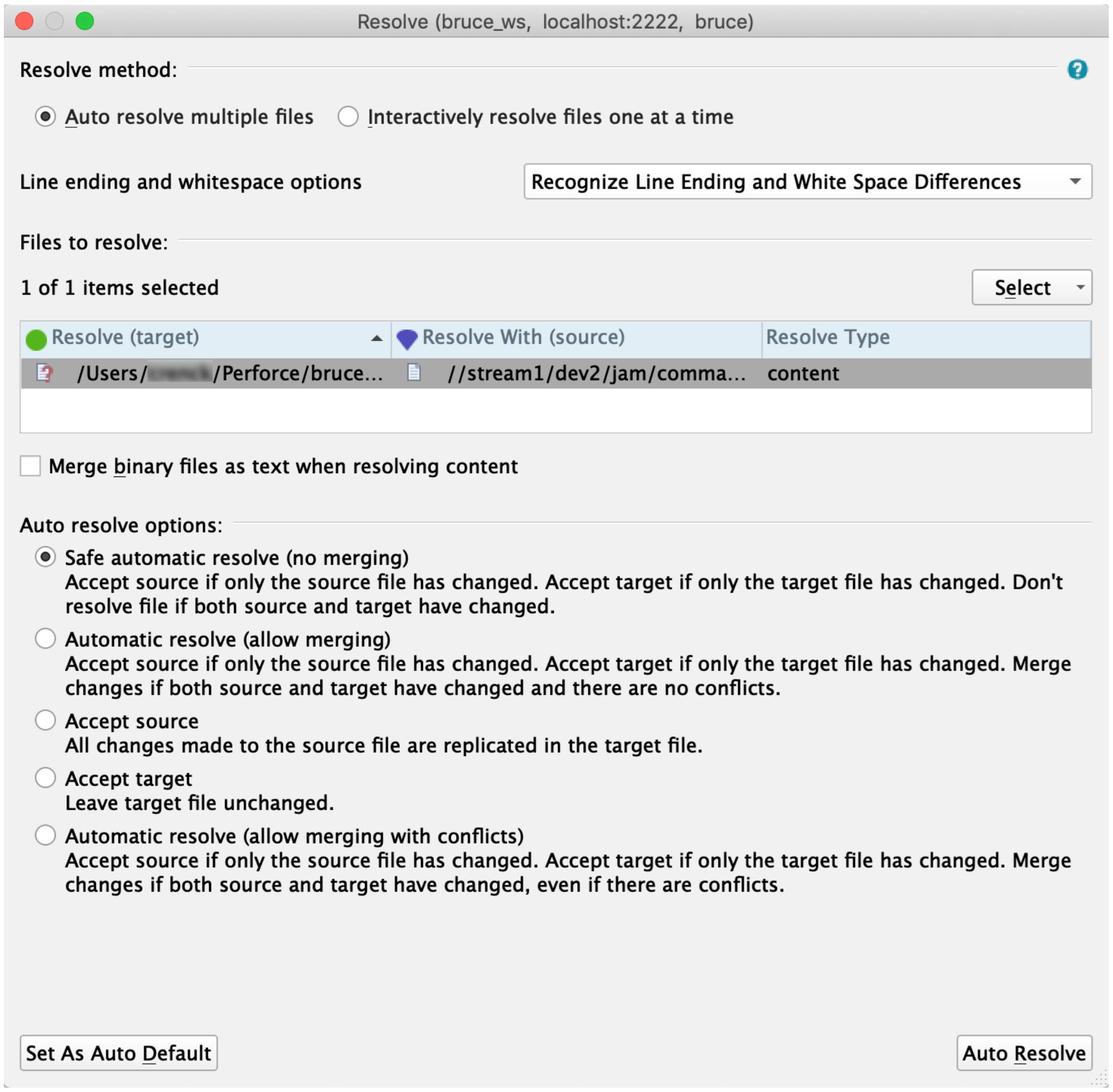
   If you select this option, P4V treats binary files like text files and attempts a textual merge between the source and target files.
6. Follow the recommended action (highlighted in blue), or select another resolve option:
   - **Accept Source**: Replaces the copy of the file in your workspace with the 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**.
7. 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, right-click, and select **Resolve**. The **Resolve** dialog opens.



Resolve (bruce_ws, localhost:2222, bruce)

Resolve method:

[✓ Auto resolve multiple files] Interactively resolve files one at a time

Line ending and whitespace options: Recognize Line Ending and White Space Differences

Files to resolve:

1 of 1 items selected

Resolve (target)	Resolve With (source)	Resolve Type
/Users/[redacted]/Perforce/bruce...	//stream1/dev2/jam/comma...	content

Merge binary files as text when resolving content

Auto resolve options:

[✓ Safe automatic resolve (no merging)]
Accept source if only the source file has changed. Accept target if only the target file has changed. Don't resolve file if both source and target have changed.

Automatic resolve (allow merging)
Accept source if only the source file has changed. Accept target if only the target file has changed. Merge changes if both source and target have changed and there are no conflicts.

Accept source
All changes made to the source file are replicated in the target file.

Accept target
Leave target file unchanged.

Automatic resolve (allow merging with conflicts)
Accept source if only the source file has changed. Accept target if only the target file has changed. Merge changes if both source and target have changed, even if there are conflicts.

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. By default, P4V recognizes line endings and white space differences. If you want P4V to ignore line endings or white spaces, select an option from the **Line ending and whitespace options** list.

   - **Ignore Line Ending Differences:** Ignores differences in line-ending convention
   - **Ignore Line Ending and White Space Length Differences:** Ignores whitespace-only changes (for example a tab replaced by eight spaces)
   - **Ignore Line Ending and All White Space Differences:** Ignores whitespace altogether (for example deletion of tabs or other whitespace)

   If you select any of these options and the files differ by whitespace only, P4V resolves the files using the text in the workspace file.

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. 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.

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

7. 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.

8. (Optional) Repeat if you are auto-resolving multiple resolve types separately.
9. 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 Core server, 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 in the *Helix Core Server 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**. 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 *Use branch mapping* as your **Merge method.**
3. Browse for the branch mapping you want to use. Select the branch mapping and click **OK**.

   The dialog refreshes to show a table with **Source** and **Target** columns and an arrow icon in between. By default, the arrow indicates the direction of the merge from source to target. You can use the option buttons labeled **source** and **target** at the bottom of the **Filter** tab to reverse the direction of the merge. If **source** is selected, files are merged from source to target. If **target** is selected, files are merged from target to source. Changing this selection also switches the direction of the arrow icon as well as the **Source** and **Target** columns.

   Important
   If you perform a merge/integrate operation from a file directory that is located at a higher level than what is specified in the branch mapping view, you see a bi-directional arrow icon instead of a directional arrow icon. This indicates that you are about to perform an advanced two-way integration: first from source to target and then from target to source. Clicking **Merge** brings up a warning that prompts you for confirmation. If you want to go back to a one-directional merge, remove the path you are filtering with from the **Merge only the following files/folders** field in the **Filter** tab. Removing the path turns the bi-directional arrow icon into a directional arrow icon. Alternatively, you can cancel the merge and select a lower-level folder for the integration.
4. To make changes to the mapping, click **Edit View**.

   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. On the **Advanced** tab, specify other options as needed:
   - **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.
   - **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.

   Warning
   The following integration flags can have unexpected or undesired results. Do not select them if you are not certain you want these actions to be applied.

    - **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 (-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).
    - **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.
    - **Check for opened files and warn prior to merging (might affect server performance):** Checks for files currently open and displays a warning before the merge.

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.

The **Description** field accepts HTML tags for marking up and hyperlinking text. For details, see "Formatting text in Description fields" on page 57.

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 45.

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** 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** 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** 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 *Helix Core 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** 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, make sure the bottom pane including the **Details** and **Files** tabs is expanded (visible). If it is not, 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. Select a label to display its associated files in the bottom pane.
3. In the bottom pane, 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**.
   - Right-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 Core server administrator. See "Customizing Perforce: Job Specifications" in the *Helix Core Server Administrator Guide*.

   The **Description** field accepts HTML tags for marking up and hyperlinking text. For details, see "Formatting text in Description fields" on page 57.

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, right-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
`word word word`	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 *all* of the words "filter", "file", and "mailbox" in any of the job fields.
`pass:[`*`word word word`*`]`	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
`^`***`word`***	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).	`filter ^file` Displays jobs that contain "filter" and do not contain "file".
`fieldname=value`	Displays jobs that include the specified value in the specified field.	`status=open owner=edk` Displays open jobs owned by edk.
`^`***`fieldname=value`***	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).	`^status=closed& subsystem=parser` Displays unclosed jobs affecting the parser subsystem.

Syntax	Description	Example
`fieldname=value+*`	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".
`yyyy/mm/dd:hh:mm:ss`	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 Core Server 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 `#`).

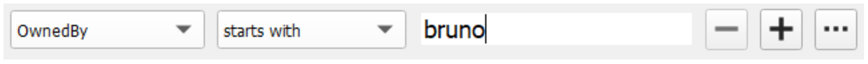
  For more information on wildcards and revision specifiers, see "Issuing P4 Commands" in the *Helix Core Server User Guide*.
- Drag and drop a file path from the Depot or Workspace tree into the field.
- Click the drop-down arrow to view and select recent file paths.

To get help constructing a file path, use the File Path Builder.

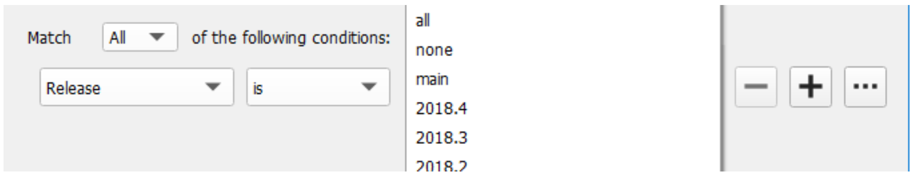
Search for jobs using the Job Query Builder

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:

     
     OwnedBy · starts with · bruno

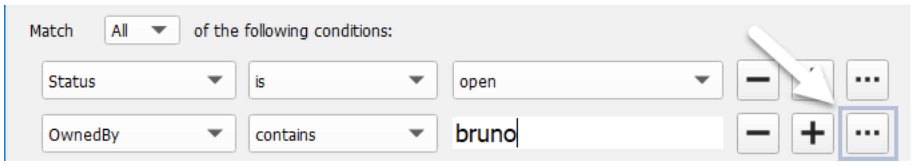
     Some values are editable; others are predefined and can be selected from a drop-down list:

     
     Match All of the following conditions:

     Release · is

     all
     none
     main
     2018.4
     2018.3
     2018.2
   - 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:



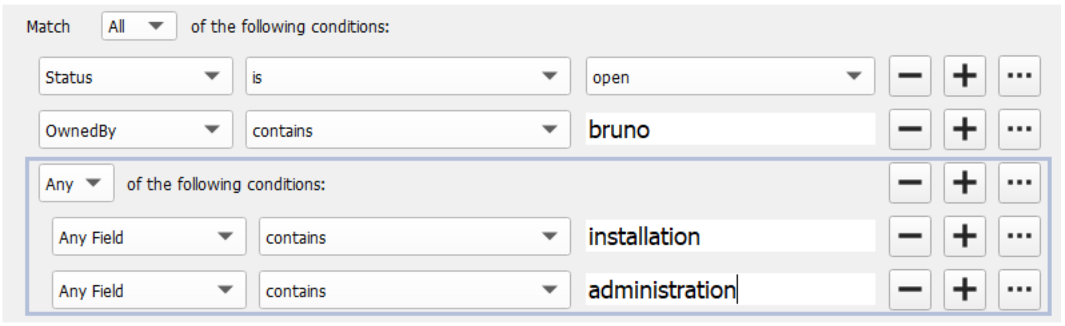
Match All of the following conditions:

Status | is | open

OwnedBy | contains | bruno

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

  - *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.*



Match All of the following conditions:

Status | is | open

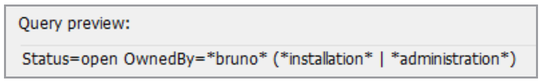
OwnedBy | contains | bruno

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 130
Environment considerations 130
Add custom tools 131
Import and export tools 133
HTML Actions 134
Run the example of pre and post submit 134
P4VJS and the examples 135
P4VJS functions for Submit 136
Replacing the Submit dialog is possible 136
To develop your own custom HTML Action 137
Launching P4V components from the command-line client, P4VC 137
Command Syntax 137

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`	*Depends on which pane has focus:* ■ 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\temp` 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.

HTML Actions

HTML Actions provide a way to customize behavior before or after a **Submit** action. For example, you might include logic that enforces a business rule regarding submissions.

You can launch an HTML page:

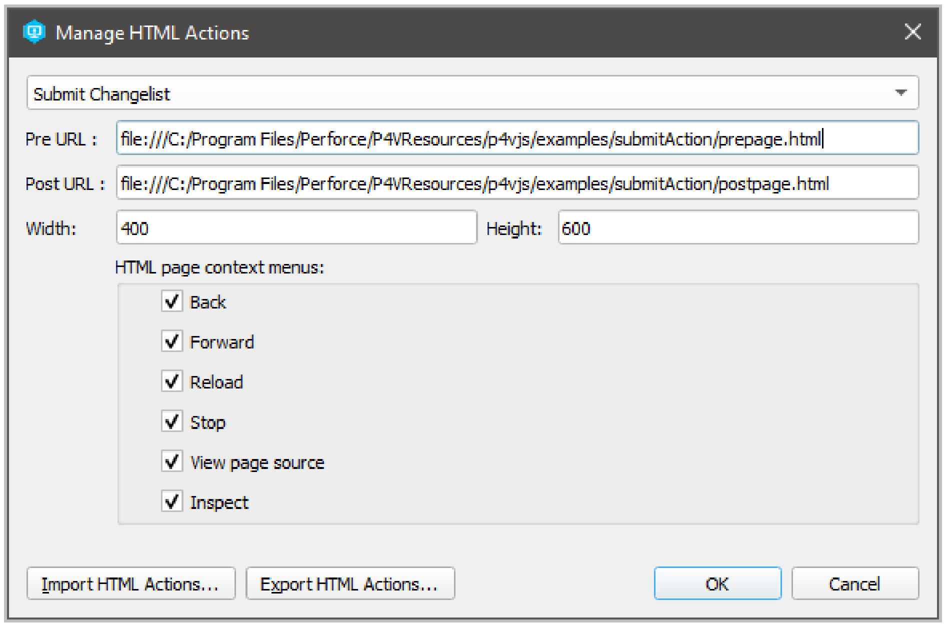
- **before** a Submit Changelist action
- **after** a successful Submit Changelist action

Run the example of pre and post submit

The two examples merely show the mechanics of an HTML Action. They might be useful as a framework for you to develop your own HTML Action in which you add the logic of your business rules around file submissions.

To run the examples:

1. In **Edit > Preferences > Tools**, check the **Enable HTML Tools** checkbox and click **OK**.
2. Restart P4V.
3. Select **Tools > Manage Tools > HTML Actions**.
4. In the **Manage HTML Actions** dialog, set the value for the **Pre URL** field and the **Post URL** field. For example,


Manage HTML Actions

Submit Changelist

Pre URL : file:///C:/Program Files/Perforce/P4VResources/p4vjs/examples/submitAction/prepage.html

Post URL : file:///C:/Program Files/Perforce/P4VResources/p4vjs/examples/submitAction/postpage.html

Width: 400 Height: 600

HTML page context menus:

- ✓ Back
- ✓ Forward
- ✓ Reload
- ✓ Stop
- ✓ View page source
- ✓ Inspect

5. Click **OK**.

In this example:

- the Submit window can display two pages:
  - the example's custom **Pre Sumit Page**
  - the P4V **Submit Changelist** page
- the **Post Submit Page** is its own window.

When the user attempts to submit, the **Pre Sumit Page** displays.

When the user clicks **Next**, the P4V **Submit Changelist** page displays.

When the user clicks **Sumit**, the **Post Submit Page** appears in its own window.

P4VJS and the examples

On Windows, assuming `C:/Program Files/Perforce/` is the installation directory, the example files are in the `P4VResources/p4vjs/examples/submitAction/` subdirectory. They are named `prepage.html` and `postpage.html`

The example `prepage.html` and `postpage.html` involve P4VJS, which is a JavaScript-based API to communicate with P4V.

If you view the source code of `prepage.html` and `postpage.html`, you will see the P4VJS function calls of the examples.

P4VJS functions for Submit

The following P4VJS functions are designed to help you customize the Submit behavior of your own HTML Action.

Function	Description	Availability
`GetURLParameter ("change")`	Returns the value of the changelist: ▪ on the `prepage.html`, this represents the changelist number of the pending change, which might be `default` or a specific integer ▪ on the `postpage.html`, this represents the changelist number of the submitted change	prepage and postpage
`GetURLParameter ("submitshelved")`	Returns `true` if the **Submit** dialog was launched from `Submit Shelved Files...` This is useful if you have a business rule that you want to enforce solely for submissions from a shelve.	prepage only
`GetURLParameter ("files")`	Returns the list of the files that are selected when the user clicks the `Submit` button from either the Workspace or the Depot Browser.	prepage only
`GetURLParameter ("directories")`	Returns the list of the directories that are selected when the user clicks the `Submit` button from either the Workspace or the Depot Browser.	prepage only
`p4vjs.nextPage()`	Shows the P4V Submit page.	prepage only
`p4vjs.closeWindow ()`	In the prepage, this closes the Submit window, which also closes the prepage window. In the postpage, this closes the postpage window.	prepage and postpage

Replacing the Submit dialog is possible

It is not likely that you will want to completely replace the standard Submit dialog, but doing so is possible. If you want to completely replace the standard P4V **Submit** dialog with a custom **Submit** dialog that you create:

1. Write a `prepage.html` that presents your custom **Submit** dialog to the user.
2. Close this `prepage.html` by calling `p4vjs.closeWindow()` without calling `p4vjs.nextPage()`, such that the standard P4V **Submit** dialog is not called.

To develop your own custom HTML Action

To develop your own custom HTML Action, see the *P4VJS User Guide* and consider making use of the P4VJS functions for Submit described above.

Launching P4V components from the command-line client, P4VC

P4VC is a command-line client that can send certain P4 command-line commands to P4V without having to open a full P4V instance.

For example, if you are a P4 command-line client user who occasionally uses P4V to view the Revision Graph or the Stream Graph, consider using P4VC.

Command Syntax

Command-line help is available:

`p4vc -h` displays the list of optons:

```
p4vc [options] command [arg ...]


Options:
    -h -?           print this message
    -V              print client version
    -c client       set client name (default $P4CLIENT)
    -C charset      set character set (default $P4CHARSET)
    -p port         set server port (default $P4PORT)
    -u user         set user's username (default $P4USER)
```

`p4vc -help` displays the list of commands.

The following table shows the output of **`p4vc help [`*`command`*`]`** for each command:

Command	Purpose	Syntax
`help`	Print the requested help message	`p4vc [options] help`
`branchmappings`	Shows list of branch mappings	`p4vc [options] branchmappings`

Command	Purpose	Syntax
`branches`	Same as branchmappings. Shows list of branch mappings	`p4vc [options] branches`
`diff`	Shows diff dialog	`p4vc [options] diff [ file (s) ]`
`diffhave`	Diff opened files against have revision (local paths)	`p4vc [options] diffhave [ file(s) ]`
`diffprev`	Diff against a previous revision (depot **path#revision**)	`p4vc [options] diffprev [ file(s) ]`
`groups`	Shows list of groups	`p4vc [options] groups`
`branch`	Show/Edit branch	`p4vc [options] branch [ spec, ... ]`
`change`	Show/Edit change	`p4vc [options] change [ spec, ... ]`
`client`	Same as workspace. Show/Edit workspace	`p4vc [options] client [ spec, ... ]`
`workspace`	Show/Edit workspace	`p4vc [options] workspace [ spec, ... ]`
`depot`	Show/Edit depot	`p4vc [options] depot [ spec, ... ]`
`group`	Show/Edit group	`p4vc [options] group [ spec, ... ]`
`job`	Show/Edit job	`p4vc [options] job [ spec, ... ]`
`label`	Show/Edit labe	`p4vc [options] label [ spec, ... ]`
`user`	Show/Edit user	`p4vc [options] user [ spec, ... ]`
`jobs`	Shows list of jobs	`p4vc [options] jobs`
`labels`	Shows list of labels	`p4vc [options] labels`
`pendingchanges`	Shows list of pending changes	`p4vc [options] pendingchanges`

Command	Purpose	Syntax
`history`	Shows file history	`p4vc [options] history [ file(s), ... ]`
`properties`	Shows file details	`p4vc [options] properties [ file(s), ... ]`
`resolve`	Launches resolve dialog	`p4vc [options] resolve [ -f ] [ -c num ] [ file(s), ... ]`
`revisiongraph`	Shows revision graph	`p4vc [options] revisiongraph [ file(s), ... ]`
`revgraph`	Same as revisiongraph. Shows revision graph	`p4vc [options] revgraph [ file(s), ... ]`
`streamgraph`	Shows stream graph pane	`p4vc [options] streamgraph [ file(s) ]`
`streams`	Shows list of streams	`p4vc [options] streams`
`submit`	Launches submit dialog	`p4vc [options] submit [ -c num ] [ file(s), ... ]`
`submittedchanges`	Shows list of submitted changes	`p4vc [options] submittedchanges`
`timelapse`	Shows timelapse view	`p4vc [options] timelapse [-l linenumber] [ file(s), ... ]`
`timelapseview`	Same as timelapse. Shows timelapse view	`p4vc [options] timelapseview [ file(s), ... ]`
`tlv`	Same as timelapse. Shows timelapse view	`p4vc [options] tlv [ file (s), ... ]`
`users`	Shows list of users	`p4vc [options] users`
`workspaces`	Shows list of workspaces	`p4vc [options] workspaces`
`clients`	Same as workspaces. Shows list of workspaces	`p4vc [options] clients`
`shutdown`	Shut downs the p4v service. This command does not require a client workspace.	`p4vc [options] shutdown`

Note

- Host = localhost Port = `7999`
- The first command sent from P4VC launches a background process. If it fails, the connection timeout is 45 seconds
- Swarm is unavailable with P4VC

8 | Working with streams

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

About streams 141
About stream views 144
Setting up streams 150
Creating streams 151
Editing streams 154
Resolving streams 157
Shelving streams 158
Streams Revision Graph 159
Selecting streams 161
Using the stream graph 161
Merging down and copying up between streams 165
Deleting stream files and streams 167
Working with task streams 168
Working with virtual streams 173

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:

- Change is propagated in a controlled way through the hierarchy that you define.
- 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.
- Some best practices, such as merge down from parent before allowing copy up to parent, are enforced automatically.

If you are new to working with the Perforce version control system, you have not yet established a robust codeline policy and associated workflow, or you do not have the staff to customize Helix Core server by writing trigger code, using streams is a good option for you. Streams come with reduced administration duties for codeline policy and workspace management, and the Stream Graph provides a graphical representation of the relationship between parent and child streams and whether a merge is required.

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 (high quality, releasable). Typically, the mainline is required to be fairly stable (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 144.

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 165.

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 streams depots through P4Admin or the P4 command-line client.

For more information, see "Setting up streams" on page 150.

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 Helix server.

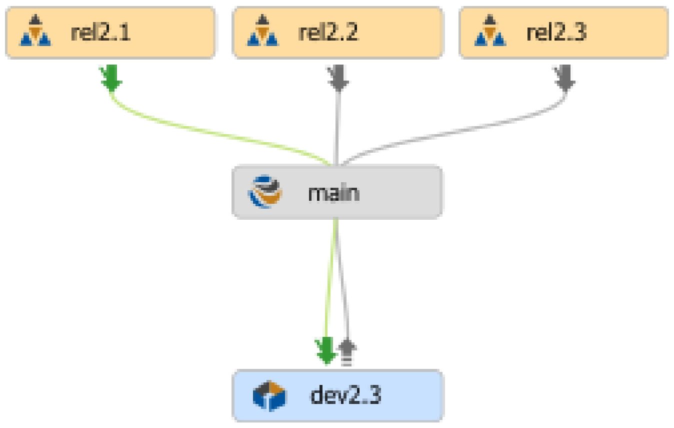
  For more information, see "Working with task streams" on page 168.
- **Virtual** streams provide users with the ability to restrict the workspace view of a real stream. Virtual streams act as a filter. They are useful when you want to:
  - Submit changes directly to a stream but do not want to sync all 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 173.

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 161.

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 (**Stream** dialog **> Advanced** tab > **Paths:** field). 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.

Tip
To search for a specific string in the stream spec, click inside the `Paths:` field and then press **Ctrl+F**. This opens the **Find** dialog.

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 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 a list of file or directory names to be ignored in workspace views, 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:
/tmp # ignores files named "tmp"
/tmp/... # ignores directories named tmp
.tmp # itnores file names ending in .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 Helix server 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 Helix server 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 streams" below.

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. Right-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. Right-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 streams

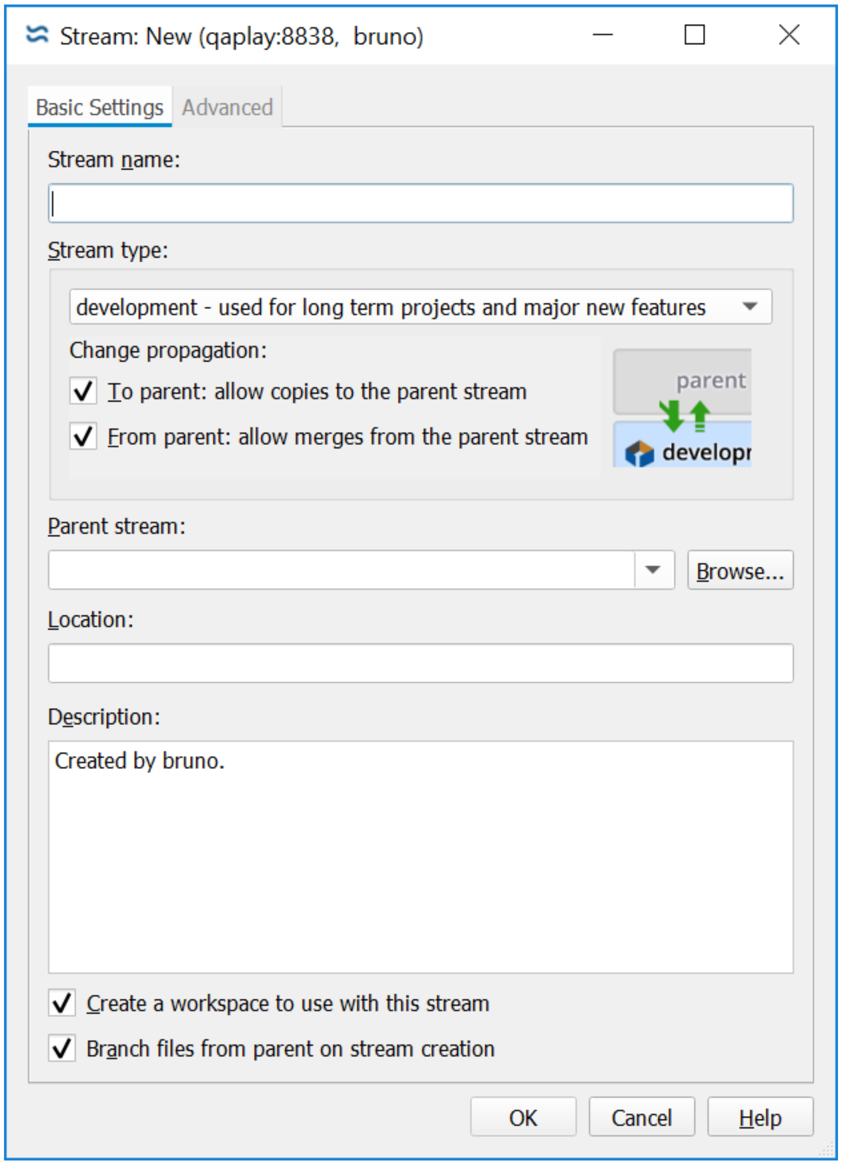
When you create a stream, you specify its properties in the **Stream: New** dialog. Here, you specify a stream depot to be treated as a stream as well as the stream's lineage, its view, and its expected flow of change.

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

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*>'...**.

   The **Stream: New** dialog opens.



Stream: New (qaplay:8838, bruno)

Basic Settings | Advanced

Stream name:

Stream type: development - used for long term projects and major new features

Change propagation:
- [✓] To parent: allow copies to the parent stream
- [✓] From parent: allow merges from the parent stream

Parent stream:

Location:

Description: Created by bruno.

- [✓] Create a workspace to use with this stream
- [✓] Branch files from parent on stream creation

2. 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, which can be any of the following:
   - Mainline: The base, trunk, or parent of a stream system.
   - Development: Used for long-term projects and major new features. Has a controlled flow. Can be changed.
   - Release: Used for fixing bugs, testing, and release distribution. Has a controlled flow. Can be changed.
   - Virtual: Used to narrow the scope and submit directly to parent.
   - Task: Used for lightweight branches, bug fixes, and new features. Task streams are usually short-lived and only promote edited files to the repository. Branches and integrated files are stored in shadow tables that are removed when the task stream is deleted or unloaded.
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**).
5. 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 the **Ignored** field 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. For example:

     ```
     /tmp # ignores files named "tmp"
     /tmp/... # ignores directories named tmp
     .tmp # itnores file names ending in .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.

6. (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.

7. (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.

8. 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" on page 161.

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.

Editing streams

You can edit a streamsin the following ways:

- **Publicly**: Any changes you save immediately propagate to everyone using the stream. Testing is basic and traceability is minimal.
- **Privately**, by checking out the stream: Any changes you make are saved in a changelist.
  The changes only become public when you check in the changes.
  Private editing of a stream allows for:

- testing changes to a stream before checking in those changes
- traceability of changes to a stream in changelists

Edit a stream publicly

Use caution when performing public edits because they could block other users. Only perform public edits when the changes to the stream configuration are straightforward and safe. When you save these changes, they immediately apply to everyone using the stream.

To edit a stream publicly:

1. In the **Streams** tab or the **Stream Graph** tab, right-click a stream, and select **Edit Stream '<*stream_name*>'**.
2. In the **Stream: Edit** window, make the required changes to the stream spec.
3. Click **Apply** to save the changes and keep the window open, or click **OK** to save the changes and close the window.

Edit a stream privately

Editing a stream privately lets you make configuration changes in isolation from other users of that stream. This is the recommended method for editing streams. This method checks out the stream to the default changelist so that you can test the changes in an isolated environment before checking them in, which makes them public.

Changelists that include a stream include a stream badge for immediate visibility, such as this one: . In this case, the changelist icon is blue because it does not contain any open files, only the checked out stream.

When all work on the stream configuration, and any associated code changes, is complete, you can submit the stream and any related files together in a single atomic changelist, which improves visibility and tracking.

The maintained history allows you to later go back and check what you changed, when you did it, and what other, related code changes you submitted at the same time.

If P4V detects any conflicts upon checkin, a question mark appears on the stream icon in the changelist and you cannot submit the changelist.

P4V provides different ways to resolve conflicts. For details, see "Resolving streams" on page 157.

To edit a stream in private:

1. Make sure that your workspace matches the stream you want to edit.
2. In the **Streams** tab or the **Stream Graph** tab, right-click a stream and do one of the following:

- To check out the stream and edit it immediately:

  Select **Check Out and Edit Stream '<*stream_name*>'**.

  P4V checks out the stream, places it in the default changelist, and opens the **Stream: Edit** window.

- To check out the stream, move it to a different changelist, and then edit it:

  a. Select **Check Out Stream '<*stream_name*>'**.

     P4V checks out the stream and places it in the default changelist.

  b. To move the stream to a different changelist, do one of the following in the **Pending** tab:

     - Drag the stream from the default changelist to the changelist of your choice.
     - Right-click the pending changelist, select **Move to Another Changelist**, and, in the **Select Pending Changelist** dialog, select the changelist that you want to move it to and click **OK**.

  c. Right-click the stream again and select **Edit Checked Out Stream '<*stream_ name*>'**.

     P4V opens the **Stream: Edit** window.

In the **Stream Graph** tab, the workspace icon for the stream turns red as an indicator that this stream is currently checked out: mainline

3. In the **Stream: Edit** window, make the required changes. For information on the available fields, see "Creating streams" on page 151.
4. Click **Apply** to save the changes and keep the window open, or click **OK** to save the changes and close the window.
5. Make sure everything is working as expected with the changes you made. If needed, fix any code in other files that needs to be adjusted. Ideally, you make those code changes against the same changelist so that you can then submit the code changes and the updated stream configuration in a single, atomic operation.
6. On the **Pending** tab, do the following:

   - If the stream icon in the changelist indicates a conflict ( ), resolve the conflict before you continue. For more information, see "Resolving streams" on the facing page.
   - If the changes look good to you and the stream icon in the changelist does not indicate any conflicts, right-click the changelist with the stream and select **Submit**.

     The **Submit Changelist** window opens. The stream is automatically selected for submit. Then continue with step 8.

   - If you are not sure about your changes and want to discard them, right-click the stream and select **Revert Stream '<*stream_name*>'**.

7. In the **Submit Changelist** window:

- If there are any files that you do not want to submit yet, clear the check boxes for those files.
- Select any other options that apply.

For more information on submitting files, see "Checking in files" on page 84.

8. Click **Submit**.

See also: "Shelving streams" on the next page.

Resolving streams

Conflicts occur when you attempt to submit a stream that another user has edited and submitted while you had the stream checked out. When a conflict occurs, Helix server schedules the stream for resolve. To indicate that a stream needs resolving, P4V displays a question mark badge .

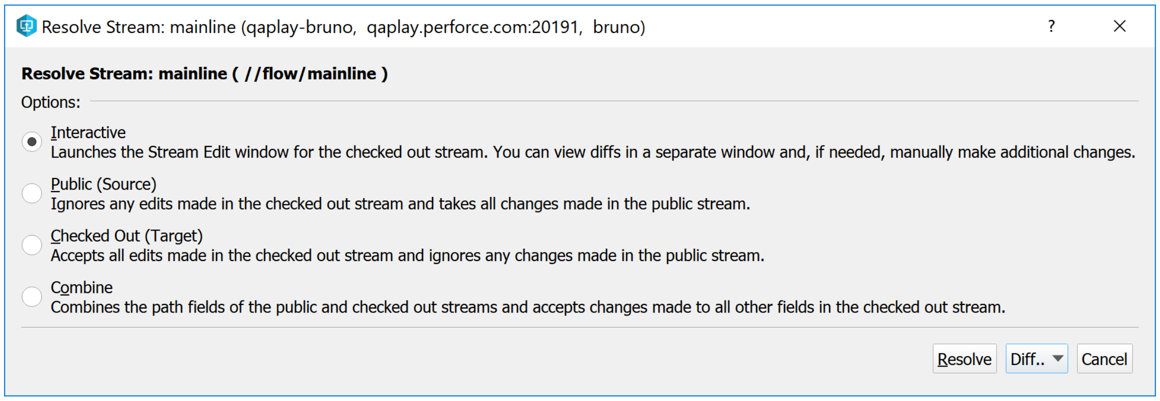
When you attempt to submit a changelist containing a stream that must be resolved, P4V displays an error message. You must resolve the changes before you can proceed.

When you are done resolving your changes, you can submit the changelist. For details, see "Editing streams" on page 154.

To resolve a stream:

1. In the **Pending** tab, right-click the stream and select **Resolve**.

   The **Resolve** dialog opens.



Resolve Stream: mainline (qaplay-bruno, qaplay.perforce.com:20191, bruno)

Resolve Stream: mainline (//flow/mainline)

Options:

(●) Interactive
Launches the Stream Edit window for the checked out stream. You can view diffs in a separate window and, if needed, manually make additional changes.

( ) Public (Source)
Ignores any edits made in the checked out stream and takes all changes made in the public stream.

( ) Checked Out (Target)
Accepts all edits made in the checked out stream and ignores any changes made in the public stream.

( ) Combine
Combines the path fields of the public and checked out streams and accepts changes made to all other fields in the checked out stream.

2. Select the resolve method:

   - **Interactive:** Default. Opens the **Stream: Edit** window. You can view diffs in a separate window and, if needed, manually make additional changes.
   - **Public (Source):** Ignores any edits made in the checked out stream and takes all changes made in the public stream.

    - **Checked Out (Target):** Accepts all edits made in the checked out stream and ignores any changes made in the public stream.
    - **Combine:** Combines the path fields of the public and checked out streams and accepts changes made to all other fields in the checked out stream.
3. Optionally, from the **Diff** list, select which versions of the stream you want to compare:
    - Base vs Checked Out (Target)
    - Public (Source) vs Checked Out (Target)
    - Base vs Public (Source)

    where:

    - The *Base* version represents the public have version from which the checked out version is derived.
    - The *Public* version represents the current public version, or source, which may have changed since the *base* version was established.
    - The *Checked Out* version represents a private copy of the *base* version. As the target version, it now includes the changes you have made.

    P4V displays the versions side by side in P4Merge.
4. Click **Resolve**.

Shelving streams

If you have chosen to "Edit a stream privately" on page 155, you also have the option to shelve the stream.

Shelving your edits to a stream allows you to:

- informally share the proposed edits with a colleague for testing and feedback before committing them to the depot
- request a more formal review through Helix Swarm. See "Integration with Swarm" on page 191.
- save work in progress
- switch the stream that you are working on

To shelve a stream, click on the pending change or checked-out stream, then right-click and select **Shelve...**

To unshelve a stream, click on the pending change or shelved stream, then right-click and select **Unshelve...**

Alternatively, you can use drag-and-drop to shelve and unshelve the stream.

To submit a shelved stream, right-click and select **Submit Shelved**, which is similar to "Submit shelved files" on page 97. Alternatively, you can submit the shelved stream together with shelved files.

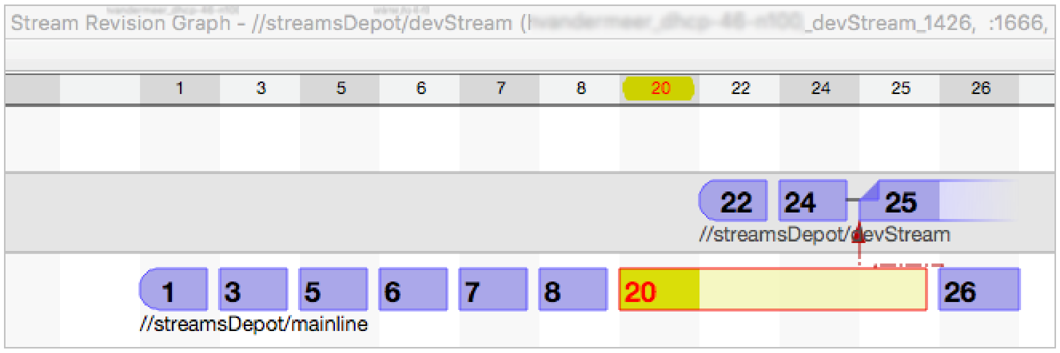
To Diff two shelved streams, drag-and-drop one stream onto the other, or use the context menu options. For example, a shelved stream can be diffed against the Have revision.

> **Note**
> A shelved stream cannot be unshelved if the stream is checked out.

Streams Revision Graph

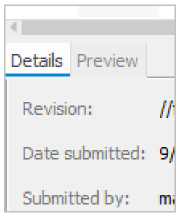
To launch the Stream Revision Graph, select a stream, right-click, and choose **Stream Revision Graph...** on the context menu.

This graph displays the changelist numbers of submitted changelists associated with edits to the stream specification.



This example is typical because:

- Although the numbers increase from left to right, the numbers are often non-consecutive. This is because most changelists are for file revision, not stream spec revisions.
- A dotted red line depicts the implicit inheritance from the latest in the parent stream to the latest in the child stream. In this case, changlist `25` in devStream inherits from changelist `26` in the mainline stream.
- The end-user has selected a specific changelist (in this case, changelist `20`), so additional data is available in the lower-left:
  - the **Details** tab shows the revision number, date submitted, submitted by, and so on
  - the **Preview** tab shows the stream spec as read-only text

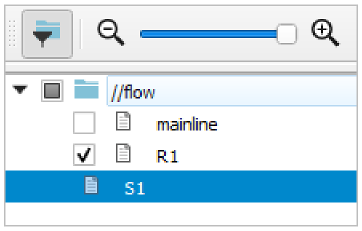
  

  These tabs are available for changelists associated with the "Edit a stream privately" on page 155 feature.

To Diff two versions of a stream spec, drag one version onto another version.

To navigate the Stream Revision Graph, use the arrow key or click a changelist number.

To filter out a stream within your model, use the funnel icon in the top-left corner and deselect that stream.



In this example, we filter out the mainline stream so we can focus on the other two streams.

Selecting streams

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

- Right-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 the Streams preferences. For more information, see "Work in a stream" on page 164.

Using the stream graph

The **Streams** tab lists streams while the **Stream Graph** tab displays a graphical representation of the relationships between parent and child streams in a selected depot.

To display streams in the Streams tab:

1. Go to **View > Streams** to open the **Streams** tab.
2. Select a display option by clicking the **Show streams in list or tree** icon :
   - **List** displays the streams in a flat list.
   - **Tree** (default) displays the streams in a hierarchy, with development streams and release streams listed under mainline streams.
3. Search for streams or restrict the streams that are displayed by using the Filter pane at the top of the tab. You can filter by owner, stream name, parent stream, stream type, and depot.

   For more information about filters, see "Searching and filtering" on page 45.

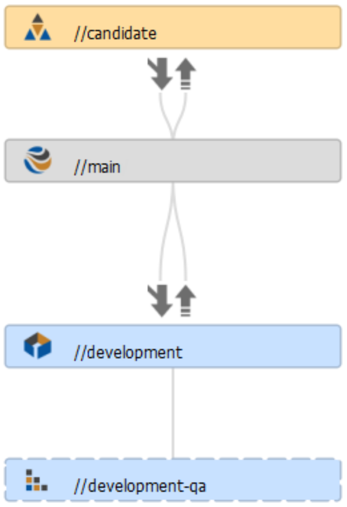
To display streams in the Stream graph:

- Go to **View > Stream Graph** to open the **Stream Graph** tab.

Stream Graph display conventions

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:

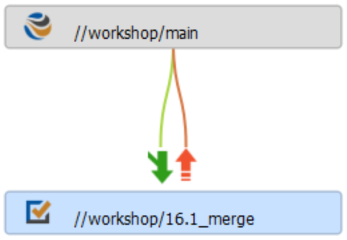
Merge indicator:

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** tab, click the arrow icon at the top left to open the filter pane.
2. From the **Depot** list, select the depot that contains the streams you want to view.
3. If the depot contains more than one mainline stream, from the **Mainline** list, select the mainline stream you want displayed in the graph. The field below the list populates with a tree view of the selected stream.

   The full depot path appears at the bottom of the filter pane and at the top of the **Stream Graph** tab, next to the arrow icon.
4. (Optional) To focus on specific child streams, select them in the tree view.

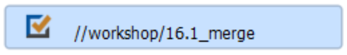
   You may need to expand the tree to view the streams you want to select. To access quick filter options, click the filter list .

   Tip
   You can select multiple streams and then press the Spacebar to select or clear the respective check boxes.
5. Click **Apply**.

   The Stream Graph displays the streams that you specified.
6. To save the Stream graph:
   a. Click the filter list and select **Save Graph As**.
   b. In the **Add Stream Graph Filter** dialog, enter a name for the filter and click **OK**.

      You can now access the filter from the filter list .
7. To close the filter pane, click the arrow icon again.
8. (Optional) To view the actual depot location of the selected streams in the Stream Graph, from the **Label** list, select **Stream Root**.

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

   
   //workshop/16.1_merge

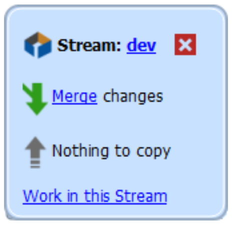
   This is useful if you have parent streams with child task streams that are located in other depots.
9. (Optional) To change the size of the stream node to accommodate long depot paths, move the **Width** slider.
10. (Optional) Click the navigator icon to open the **Graph Navigator**.

    The Graph Navigator allows you to navigate the Stream graph when it is too big to fit into the available display area. In the **Graph Navigator** window, the red rectangle indicates the portion of the graph that is currently displayed in the Stream graph. You can use your mouse or cursor keys to move the rectangle and shift focus in the Stream graph.

Note
The navigator icon is only available if the Stream graph is larger than the display area.

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:

- In the **Streams** tab or the **Stream Graph**, right-click a stream and select **Work in this Stream**.
- In the **Stream Graph**, drag the Workspace icon from your current stream to the one you want to work in.

If your stream preferences are set to use a different workspace and to show an information dialog when switching workspaces, P4V prompts you to switch workspaces or create a new workspace. In this case, click the **Create New Workspace** button (if you have only one workspace), **Switch Workspaces** button (if you have two workspace), or the **Select Workspaces** button (if you have more than two workspace) 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.

If your preferences are set to use the same workspace when you switch between streams, 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. When you do this and your Helix server version is 2019.1 or later, then if you have files checked out to:

- **The default changelist:** P4V shelves them automatically in a numbered changelist with the description "Switched branch shelf" and unshelves them again to the default changelist when you switch back to that stream. However, note that the numbered changelist created for the interim shelf does not get deleted and stays behind in the system.
- **A numbered changelist:**P4V prompts you to shelve those files before switching. When you switch back to that stream, those files remain shelved in the numbered changelist. You have to unshelve them manually.

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, right-click the stream and select **Create New Stream from *'stream_ name*'...**.
2. In the **Stream: New** dialog, define the new stream.

   See "Creating streams" on page 151.
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 (list or tree view) or the Stream Graph Tab, right-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:

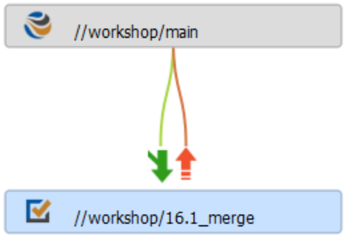
Merge indicator: ↓

Copy indicator: ↑

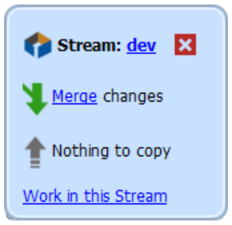
The arrows are color-coded to indicate status:

- **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, right-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 110.

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, right-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.

Deleting stream files and streams

Deleting a stream involves deleting the files and deleting the stream spec in two consecutive steps. P4V executes both steps when you select to delete a stream, but technically, you can do one without the other. If you delete only the files in a stream, the stream spec remains in the system. If you delete only the stream spec, the files that were associated with that stream remain available in the depot.

You can only delete the files in a stream if:

- You are the owner of the stream.
- The stream is a development or release stream.

P4V removes unmodified files from the stream but does not delete files with more than one revision so that their edit history is preserved. Files deleted from the stream remain in client workspaces until the next sync operation (**Get Latest Revision** option), which removes them. If deleted files have been branched to a child stream, P4V generates new integration records to directly link the branched files in the child stream to the files in the parent stream that they were previously indirectly related to.

Subsequently, P4V removes the stream spec, but only if the stream:

- Does not have any children.
- Is not referenced by any workspace.

Note
With task streams, the unmodified files automatically go away when the stream spec does. For more information, see "Delete and unload task streams" on page 172.

To delete a stream:

1. In the **Streams** view, right-click the stream you want to delete and select **Delete Stream <*stream name*>**.
2. When prompted if you want to permanently delete unmodified files before deleting the stream:
   a. Optionally, to also delete files that have been merged down multiple times from the parent stream but have never been edited in this stream, select the check box.
   b. Take note of the number of integration and revision records this action would delete.
   c. Click **Yes, delete files**.

      P4V permanently removes unmodified files (files with one revision) from this stream.
3. In the **Delete Stream** form, click **Yes** to confirm.

   P4V removes the stream spec.

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 Helix server. 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 or use Diff against those files.
- 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 171.

- 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 next 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, right-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 streams" on page 151 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, right-click an existing stream and select **Create New Stream from '*stream_name*'**.
   - In the Streams tab, right-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 streams" on page 151 for more information about the Stream:New dialog.

3. Refresh the Stream Graph view to verify that the new task stream appears correctly.

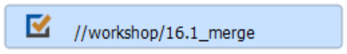
   - In the child stream’s depot, the parent stream and parent stream’s depot appear grayed out under **Graph View Options**, and the task stream appears as normal.

     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:



//workshop/16.1_merge

- 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, right-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, right-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 streams" on page 151 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 right-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 right-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 109.

  You can also populate your new task stream using the Merge/Integrate dialog. For more information, see "Merging files between codelines" on page 110.

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. Right-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. Right-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. Right-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. Right-click the stream and select **Reload Stream**.

For more information about unloading, see the *Helix Core 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:

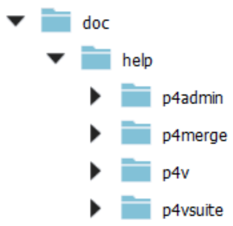
1. Go to **P4V > Preferences**(Mac) or **Edit > Preferences** (Windows) and select **Files and History**.
2. Select **Hide files/revisions from 'task' streams (when following branch, copy actions)**.

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 *help* 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 help 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 doc/help/p4admin/...
   ```

This gives you a virtual stream with the workspace view you want:

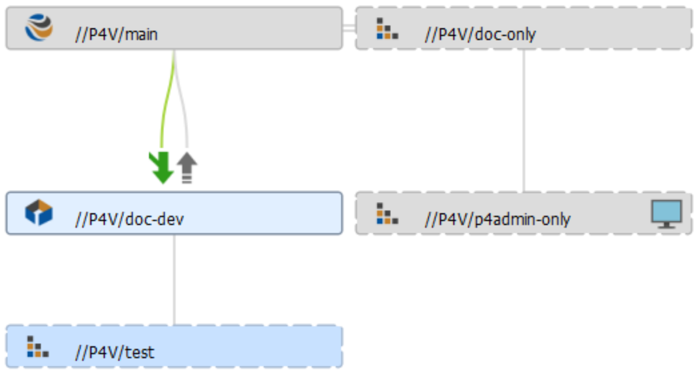
```
//P4V/doc-dev/doc/help/p4admin/... doc/help/p4admin/...
```

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 Core Server 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,** right-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

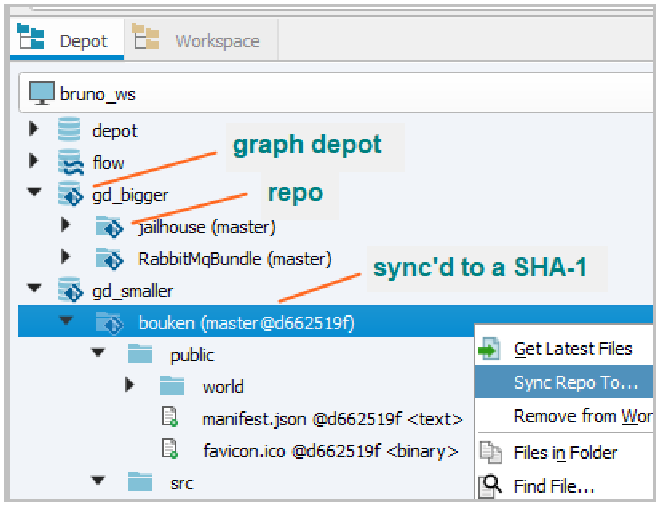
You merge changes down or copy changes up to a virtual stream just as you would to a real stream, however the merge or copy is actually submitted to the virtual stream’s base parent. You might find it more straightforward to copy and merge directly between real streams.

9 | Working with graph depots

P4V version 2020.1 is the first version to support workspaces of map to:

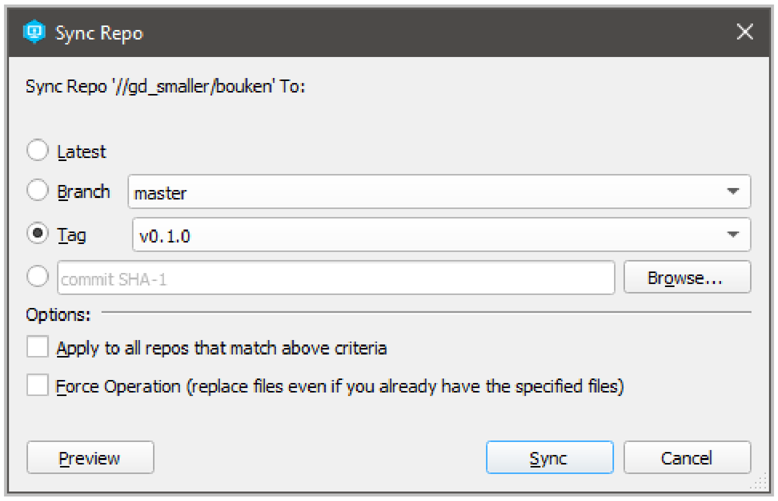
- one or more repos in one or more graph depots - see Depots and repos in the *Helix4Git Administrator Guide*
- one or more hybrid workspaces. - see "Hybrid client that maps to both classic and graph depots" in p4 client (graph) in *Helix Core P4 Command Reference*

The following image shows an example of how P4V displays a **graph depot** that contains a **repo** that contains folders and files. Each repo indicates its current branch in parentheses, such as `(master)`. Each file is identified with a **SHA-1** hash that displays in the short form of 8 hexidecimal digits:



The P4V administrator can create a graph depot and assign access permissions to graph depots or repos . See *P4Admin User Guide*.

If your P4V administrator has granted you access, you can right-click a repo, and click **Sync Repo To...**. The **Sync Repo** dialog appears, and you can sync to Latest, a Branch, a Tag, or a Commit SHA-1.

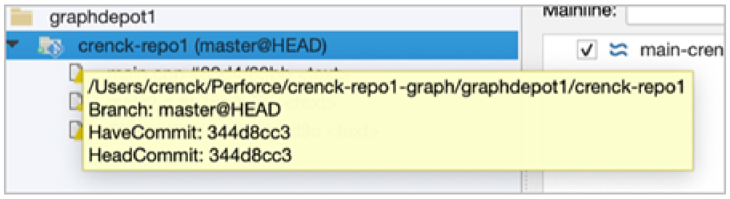

Sync Repo
Sync Repo '//gd_smaller/bouken' To:
Latest
Branch
master
Tag
v0.1.0
commit SHA-1
Browse...
Options:
Apply to all repos that match above criteria
Force Operation (replace files even if you already have the specified files)
Preview
Sync
Cancel

Important
In P4V version 2020.1,

- a graph depot is read-only
- if a hybrid workspace is of type `writeable`, the graph depot files are read-only
- if a hybrid workspace is of type `graph`, all the files in the workspace are read-only

Repo folder information

If you sync a repo to a tag or a SHA-1, the repo folder shows `branch@HEAD` or `branch@haveCommit`:



```
/Users/crenck/Perforce/crenck-repo1-graph/graphdepot1/crenck-repo1
Branch: master@HEAD
HaveCommit: 344d8cc3
HeadCommit: 344d8cc3
```

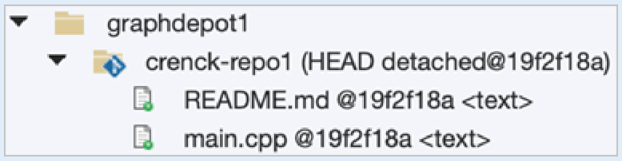
and the tooltip indicates @HEAD corresponds to the SHA-1 for both HaveCommit and HeadCommit.

Note
For details about repo file information in Helix Core server, see the output of `p4 help-graph fstat`

Tag names, if any, appear in the repo tooltip.

A branch can be `branchName@` or `Head detached@`

Note
If you sync to a tag or a SHA-1 that is not an ancestor of your current branch, `(HEAD detached@`*`a1b2c3d4`*`)` appears to indicated that the repo is not sync'd to the latest state (HEAD) of any branch. For example,



```
▼ graphdepot1
  ▼ crenck-repo1 (HEAD detached@19f2f18a)
      README.md @19f2f18a <text>
      main.cpp @19f2f18a <text>
```

Repo file information

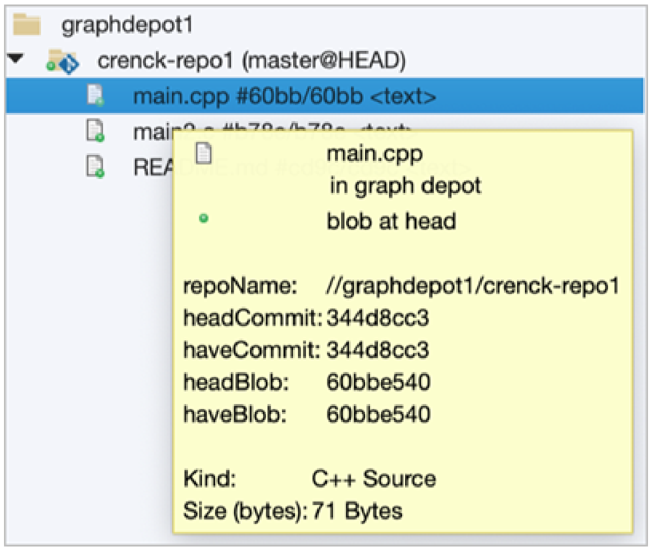
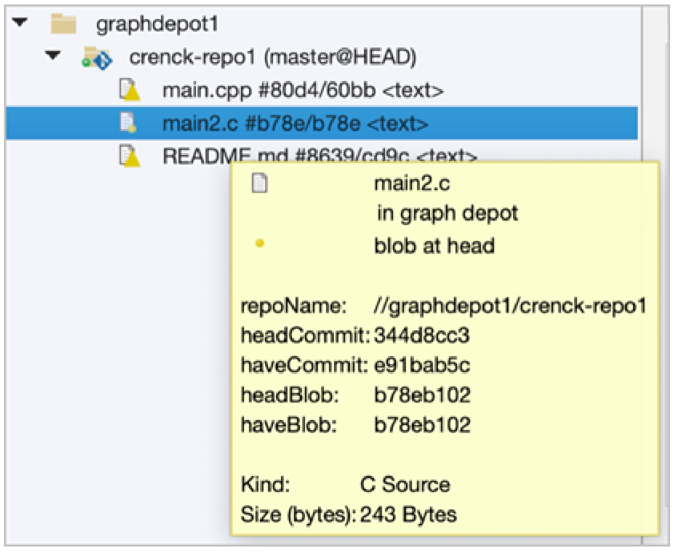
A repo file is displayed like this:

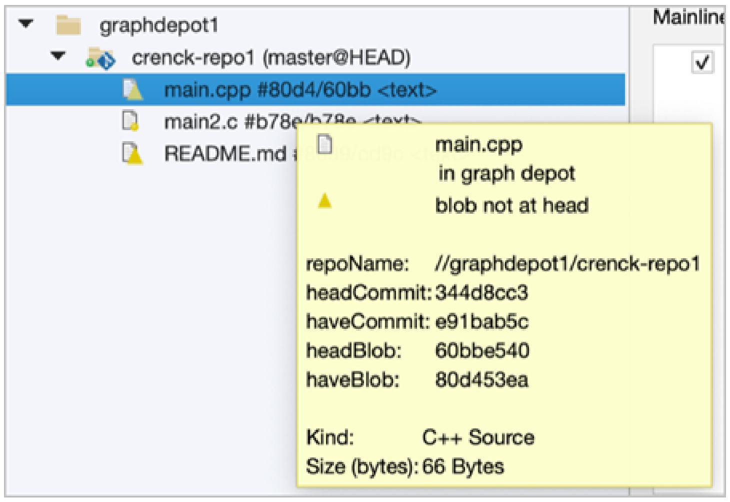
`filename.txt #haveBlob/headBlob`

where a file is represented as a binary large object (`blob`):

Repo	Client
`headBlob` represents the SHA-1 for the file content at the head revision	`haveBlob` represents the SHA-1 for the file content at the have revision on the client
`headCommit` represents the commit SHA-1 for the file at the head revision	`haveCommit` represents the commit SHA-1 for the file at the have revision on the client

In the depot tree tab, if file shows `#none/headBlob`, it means the file exists in the depot but it's not currently synced based on current tag or SHA-1.

means that the repo file is both at **`headBlob`** and at **`headCommit`**, such that ▪ **`haveBlob`** matches **`headBlob`** and ▪ **`haveCommit`** matches **`headCommit`**:	 graphdepot1 crenck-repo1 (master@HEAD) main.cpp #60bb/60bb <text> mai REA main.cpp in graph depot blob at head repoName: //graphdepot1/crenck-repo1 headCommit: 344d8cc3 haveCommit: 344d8cc3 headBlob: 60bbe540 haveBlob: 60bbe540 Kind: C++ Source Size (bytes): 71 Bytes
means that the repo file is at **`headBlob`** but not at **`headCommit`**, such that ▪ **`haveBlob`** matches **`headBlob`** but ▪ **`haveCommit`** does not match **`headCommit`** In this case, just as with a classic file, you must sync to get latest before submitting.	 graphdepot1 crenck-repo1 (master@HEAD) main.cpp #80d4/60bb <text> main2.c #b78e/b78e <text> README.md #8639/cd9c <text> main2.c in graph depot blob at head repoName: //graphdepot1/crenck-repo1 headCommit: 344d8cc3 haveCommit: e91bab5c headBlob: b78eb102 haveBlob: b78eb102 Kind: C Source Size (bytes): 243 Bytes

means that the repo file is not at **`headBlob`** and not at **`headCommit`**, such that ▪ **`haveBlob`** does not match the**`headBlob`** and ▪ **`haveCommit`** does not matche **`headCommit`**	

graphdepot1
crenck-repo1 (master@HEAD)
main.cpp #80d4/60bb <text>
main2.c #b78e/b78e <text>
README.md

main.cpp
in graph depot
blob not at head

```
repoName:    //graphdepot1/crenck-repo1
headCommit: 344d8cc3
haveCommit: e91bab5c
headBlob:    60bbe540
haveBlob:    80d453ea

Kind:          C++ Source
Size (bytes): 66 Bytes
```

10 | Using P4V for distributed versioning

This chapter provides an introduction to Distributed Versioning via P4V.

Understanding DVCS and setting up the server 183
Init 183
Read this first 184
Directories and files 184
Add files 185
Clone 185
Prepare to fetch and push content between servers 186
Fetch and push 186
Configure security for fetching and pushing 187
Specify what to copy 187
What do Fetch and Push copy? 187
Resubmit 187
Branches 188
Understanding remote mappings 189

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 Server for Distributed Versioning* for information on the distributed model.

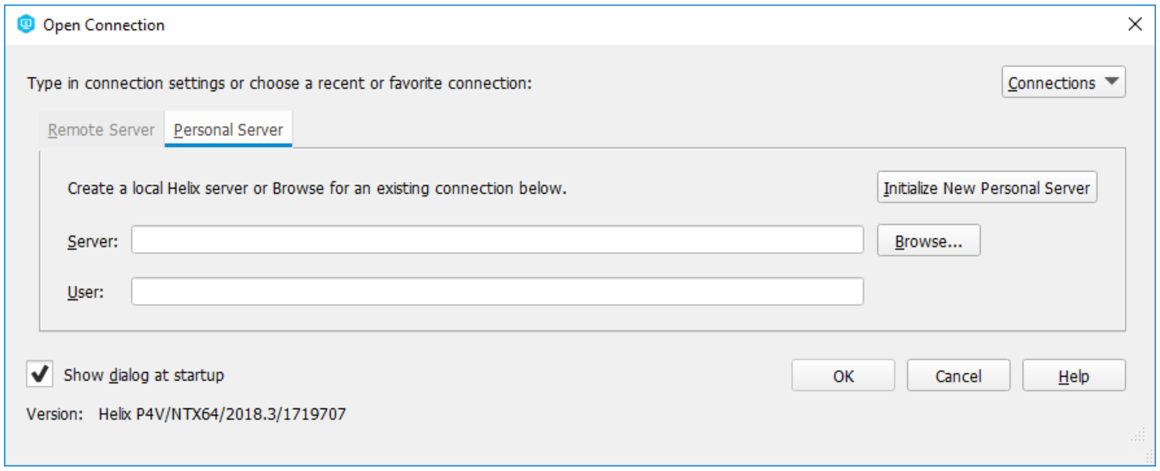
The DVCS functionality of P4V allows users to create personal servers and submit changes locally without the need to connect or set up a remote server. If the user wishes to clone, fetch or push assets from a remote server, that server must meet the minimum version of 2016.2.1487173 and be set up to 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**.


Open Connection
Type in connection settings or choose a recent or favorite connection:
Connections
Remote Server
Personal Server
Create a local Helix server or Browse for an existing connection below.
Initialize New Personal Server
Server:
Browse...
User:
Show dialog at startup
OK
Cancel
Help
Version: Helix P4V/NTX64/2018.3/1719707

Before you continue, go to "Read this first" below.

For the command-line equivalent of `Init`, see Initializing a Server in *Using Helix Core Server 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 80.

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 189 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.

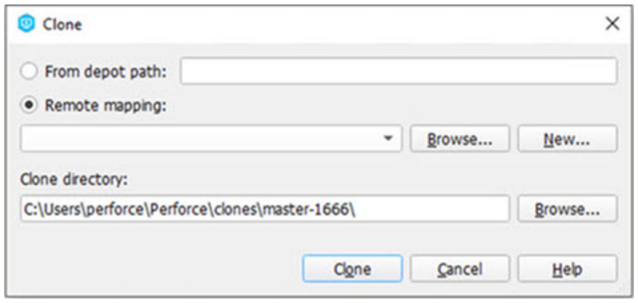
You can perform a Clone operation from a remote server using any of the following methods:

- 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.



Clone directory: C:\Users\perforce\Perforce\clones\master-1666\

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 189 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 189.

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 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 requires user interaction.

The **Resubmit** dialog opens if the fetch has conflicts. At the top is a list of all affected changelists, along with their resubmit status. Below that list is the current changelist that is being resolved.

You can interrupt the resolve/resubmit cycle at any time by clicking **Finish Later**. It is highly recommended that you complete the cycle before performing any actions to affected files to minimize potential data loss. After a restart, P4V reopens the **Resubmit** dialog.

If a file needs to be resolved before it is resubmitted, the **Resubmit** dialog provides an option to resolve the file conflict. The P4V**Resolve** dialog loads prepopulated with the files associated in the current step of the resolve/resubmit cycle.

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. The tangent depot is a system-generated, read-only location in which files with conflicting changes are stored. Files are automatically moved to the tangent depot as part of the Fetch process. For more information on the various kinds of depots, including the tangent depot, see the `p4 depot` chapter in the *Helix Core P4 Command Reference*. 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 is 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.

For more information, see `p4 resubmit` in the *Helix Core P4 Command Reference* or run the `p4 help resubmit` command from the command line (you can open the command window by selecting **File > Open Command Window Here**).

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 141. 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 161.

To branch in P4V, from the Stream Graph View, right-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

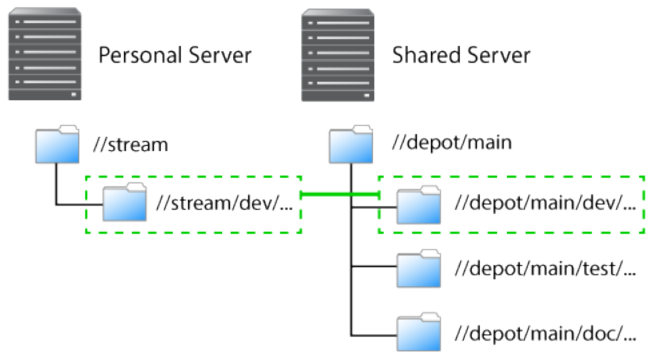
Note You cannot switch to a new branch if files are open in a numbered changelist. If files are open in the default changelist, they will be shelved and reverted prior to switching to the new branch, and will be automatically unshelved when switching back to thi

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.

11 | 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.

Note
A Swarm review can be for files, a stream spec, or a stream spec along with files. See "Shelving streams" on page 158.

Swarm can be customized 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*.

Review workflow

Following is the happy path workflow for a Swarm review. For more information, 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 metadata, 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, they can change the state of a review to **Needs Revision**, which notifies the author.
5. **Request further review**: Authors can request a review of the revision 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. When they approve or reject a review, the review is considered closed.

7. **Commit the review**: For pre-commit reviews, authors can commit reviews using either:
   - Swarm
   - their Helix server clients, such as P4V or P4VS. In this case, 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 Core Server Administrator Guide*.

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 might want 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

When Swarm integration is enabled, a number of additional features is available in P4V, including new context menus, request review and update review dialogs, badging on pending changes, committed changes, and history, 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.

Reviewers:

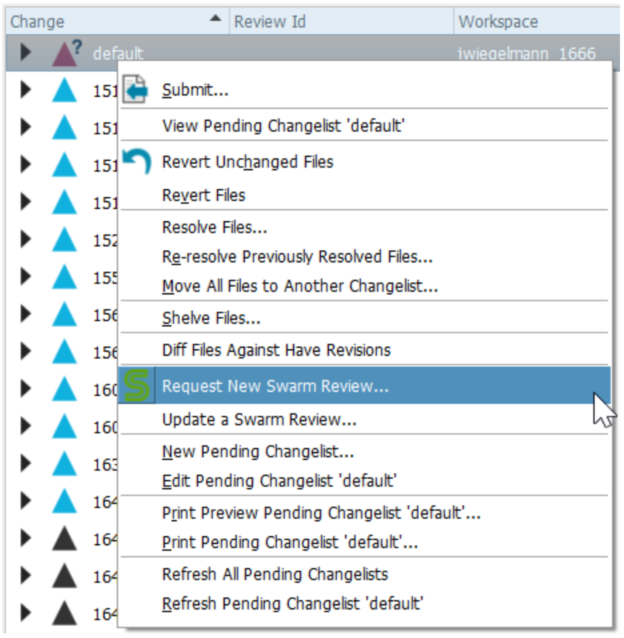
- When you select reviewers, you can make them *required* or *optional*.
- When you select a group as reviewer, you can make all members optional, all members required, or decide that only one member of the group is required to review.
- If a user is specified as an optional reviewer, but the same user is also the member of a group of which all members are required to review, then this user is actually a required reviewer.

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**
   > If the changelist is already part of a Swarm review, this option is not available.

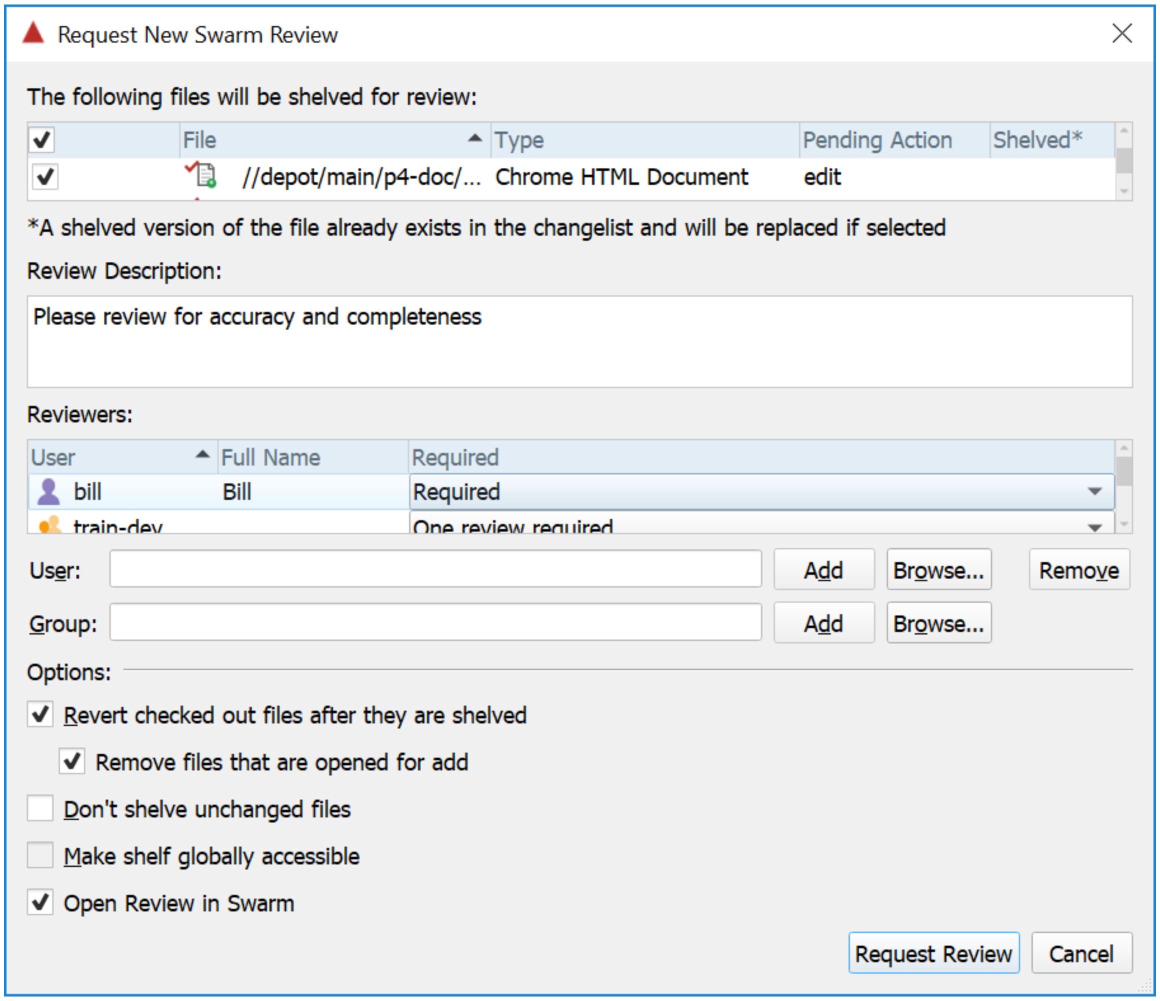

Change
Review Id
Workspace
default
iwiegelmann_1666
Submit...
View Pending Changelist 'default'
Revert Unchanged Files
Revert Files
Resolve Files...
Re-resolve Previously Resolved Files...
Move All Files to Another Changelist...
Shelve Files...
Diff Files Against Have Revisions
Request New Swarm Review...
Update a Swarm Review...
New Pending Changelist...
Edit Pending Changelist 'default'
Print Preview Pending Changelist 'default'...
Print Pending Changelist 'default'...
Refresh All Pending Changelists
Refresh Pending Changelist 'default'

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 (users or groups), 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	Type	Pending Action	Shelved*
✓	//depot/main/p4-doc/...	Chrome HTML Document	edit	

*A shelved version of the file already exists in the changelist and will be replaced if selected

Review Description:

Please review for accuracy and completeness

Reviewers:

User	Full Name	Required
bill	Bill	Required
train-dev		One review required

User: Add Browse... Remove

Group: Add Browse...

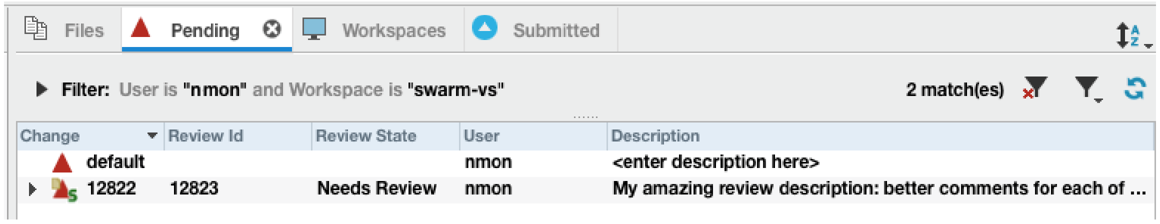
Options:

- ✓ Revert checked out files after they are shelved
  - ✓ Remove files that are opened for add
- ☐ Don't shelve unchanged files
- ☐ Make shelf globally accessible
- ✓ Open Review in Swarm

Request Review Cancel

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.



Files Pending Workspaces Submitted

Filter: User is "nmon" and Workspace is "swarm-vs" 2 match(es)

Change	Review Id	Review State	User	Description
default			nmon	<enter description here>
12822	12823	Needs Review	nmon	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
   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.
3. Optionally, select additional options, including reviewers and opening the 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.

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.
3. Optionally, select additional options, including reverting checked out files after they are shelved,

not shelving unchanged files, and opening the review in Swarm.

4. 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:

1. 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.
2. In the **Update Review** field, enter the ID of the review you would like to update with these files.
3. Optionally, to validate if this review is in fact the review you would like to update, click **View Review Description**.
4. Optionally, select additional options, including reverting checked out files after they are shelved, not shelving unchanged files, and opening the review in Swarm.
5. 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.
2. In the **Update Review** field, enter the ID of the review to which you would like to add these files.
3. Optionally, to validate if this review is in fact the review to which you would like to add these files, click **View Review Description**.
4. Optionally, select additional options.
5. Click **Add to Review**.

   The files are 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 confirms that the review has been created or updated.

> **Note**
> If the server you are connecting to is configured for authentication with Helix Authentication Service, the Identity Provider (IdP) web page opens, prompting you for the credentials registered with your Identity Provider (IdP). For details, see https://github.com/perforce/helix-authentication-service or contact your Helix Core server administrator.

> **Note**
> If Swarm is configured for authentication with Helix Authentication Service or Helix SAML, you might need to log in again.

Getting Files from a Review

You might be inspecting a review in Swarm and decide you want a local copy of the files. The review ID shown in Swarm corresponds to a pending changelist in Helix server that contains these files. To get these files from P4V:

1. In P4V, open the **Go To Spec** dialog with the keyboard shortcut of **CTRL+F** or **COMMAND-G** on Mac.
2. Enter the review ID and click **OK**.

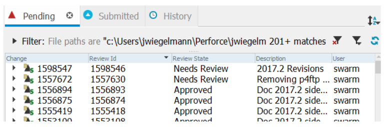
   P4V displays the **Pending Changelist** dialog.
3. Right-click the files you want to unshelve and select **Unshelve**.

   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.

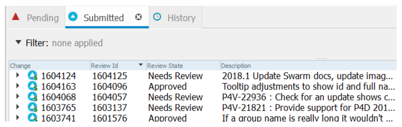
Review ID and Review State columns

P4V adds a **Review Id** and **Review State** column to the **Submitted** and **Pending** tabs and the **History** tab for connections that have the Swarm integration enabled.



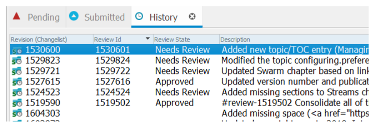
Filter: File paths are "c:\Users\jwiegelmann\Perforce\jwiegelm 201+ matches

Change	Review Id	Review State	Description	User
1598547	1598546	Needs Review	2017.2 Revisions	swarm
1557672	1557630	Needs Review	Removing p4ftp ...	swarm
1556894	1556893	Approved	Doc 2017.2 side...	swarm
1556875	1556874	Approved	Doc 2017.2 side...	swarm
1555419	1555418	Approved	Doc 2017.2 side...	swarm



Filter: none applied

Change	Review Id	Review State	Description
1604124	1604125	Needs Review	2018.1 Update Swarm docs, update imag...
1604163	1604096	Approved	Tooltip adjustments to show id and full na...
1604068	1604057	Needs Review	P4V-22936 : Check for an update shows c...
1603765	1603137	Needs Review	P4V-21821 : Provide support for P4D 201...
1603741	1601576	Approved	If a group name is really long it wouldn't ...



Revision (Changelist)	Review Id	Review State	Description
1530600	1530601	Needs Review	Added new topic/TOC entry (Managin
1529823	1529824	Needs Review	Modified the topic configuring.prefere
1529721	1529722	Needs Review	Updated Swarm chapter based on link
1527615	1527616	Approved	Updated version number and publicat
1524523	1524524	Needs Review	Added missing sections to Streams ch
1519590	1519502	Approved	#review-1519502 Consolidate all of t
1604303			Added missing space (<a href="https

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.

Reconnect to Swarm

If the connection to Swarm is interrupted, you can re-establish it from the **Connection > Reconnect to Swarm** menu.

Note
This menu item is only available if the `P4.Swarm.URL` is defined.

Glossary

A

access level

A permission assigned to a user to control which commands the user can execute. See also the 'protections' entry in this glossary and the 'p4 protect' command in the P4 Command Reference.

admin access

An access level that gives the user permission to privileged commands, usually super privileges.

APC

The Alternative PHP Cache, a free, open, and robust framework for caching and optimizing PHP intermediate code.

archive

1. For replication, versioned files (as opposed to database metadata). 2. For the 'p4 archive' command, a special depot in which to copy the server data (versioned files and metadata).

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.

avatar

A visual representation of a Swarm user or group. Avatars are used in Swarm to show involvement in or ownership of projects, groups, changelists, reviews, comments, etc. See also the "Gravatar" entry in this glossary.

B

base

For files: The file revision, in conjunction with the source revision, used to help determine what integration changes should be applied to the target revision. For checked out streams: The public have version from which the checked out version is derived.

binary file type

A Helix server file type assigned to a non-text file. By default, the contents of each revision are stored in full, and file revision is stored in compressed format.

branch

(noun) 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 to that location. A group of related files is often referred to as a codeline. (verb) To create a codeline by copying another codeline with the 'p4 integrate', 'p4 copy', or 'p4 populate' command.

branch form

The form that appears when you use the 'p4 branch' command to create or modify a branch specification.

branch mapping

Specifies how a branch is to be created or integrated by defining the location, the files, and the exclusions of the original codeline and the target codeline. The branch mapping is used by the integration process to create and update branches.

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. This is the same as branch mapping.

broker

Helix Broker, a server process that intercepts commands to the Helix server and is able to run scripts on the commands before sending them to the Helix server.

C

change review

The process of sending email to users who have registered their interest in changelists that include specified files in the depot.

changelist

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 Helix server. The changes specified in the changelist are not stored in the depot until the changelist is submitted to the depot. See also atomic change transaction and changelist number.

changelist form

The form that appears when you modify a changelist using the 'p4 change' command.

changelist number

An integer that identifies a changelist. Submitted changelist numbers are ordinal (increasing), but not necessarily consecutive. For example, 103, 105, 108, 109. A pending changelist number might be assigned a different value upon submission.

check in

To submit a file to the Helix server depot.

check out

To designate one or more files, or a stream, for edit.

checkpoint

A backup copy of the underlying metadata at a particular moment in time. A checkpoint can recreate db.user, db.protect, and other db.* files. See also metadata.

classic depot

A repository of Helix server files that is not streams-based. Uses the Perforce file revision model, not the graph model. The default depot name is depot. See also default depot, stream depot, and graph depot.

client form

The form you use to define a client workspace, such as with the 'p4 client' or 'p4 workspace' commands.

client name

A name that uniquely identifies the current client workspace. Client workspaces, labels, and branch specifications cannot share the same name.

client root

The topmost (root) directory of a client workspace. If two or more client workspaces are located on one machine, they should not share a client 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, but it can be overridden. Client workspaces, labels, and branch specifications cannot share the same name.

code review

A process in Helix Swarm by 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

Feedback provided in Helix Swarm on a changelist, review, job, or a file within a changelist or review.

commit server

A server that is part of an edge/commit system that processes submitted files (checkins), global workspaces, and promoted shelves.

conflict

1. A situation where two users open the same file for edit. One user submits the file, after which the other user cannot submit unless the file is resolved. 2. A resolve where the same line is changed when merging one file into another. This type of conflict occurs when the comparison of two files to a base yields different results, indicating that the files have been changed in different ways. In this case, the merge cannot be done automatically and must be resolved manually. See file conflict.

copy up

A Helix server best practice to copy (and not merge) changes from less stable lines to more stable lines. See also merge.

counter

A numeric variable used to track variables such as changelists, checkpoints, and reviews.

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

default changelist

The changelist used by a file add, edit, or delete, unless a numbered changelist is specified. A default pending changelist is created automatically when a file is opened for edit.

deleted file

In Helix server, a file with its head revision marked as deleted. Older revisions of the file are still available. in Helix server, a deleted file is simply another revision of the file.

delta

The differences between two files.

depot

A file repository hosted on the server. A depot is the top-level unit of storage for versioned files (depot files or source files) within a Helix Core server. It contains all versions of all files ever submitted to the depot. There can be multiple depots on a single installation.

depot root

The topmost (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. Depot syntax begins with: //depot/

diff

(noun) A set of lines that do not match when two files, or stream versions, are compared. A conflict is a pair of unequal diffs between each of two files and a base, or between two versions of a stream. (verb) To compare the contents of files or file revisions, or of stream versions. See also conflict.

donor file

The file from which changes are taken when propagating changes from one file to another.

E

edge server

A replica server that is part of an edge/commit system that is able to process most read/write commands, including 'p4 integrate', and also deliver versioned files (depot files).

exclusionary access

A permission that denies access to the specified files.

exclusionary mapping

A view mapping that excludes specific files or directories.

extension

Similar to a trigger, but more modern. See "Helix Core Server Administrator Guide" on "Extensions".

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, which 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, which is 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, preceded by a pound sign (#), for example testfile#3.

file tree

All the subdirectories and files under a given root directory.

file type

An attribute that determines how Helix server stores and diffs a particular file. Examples of file types are text and binary.

fix

A job that has been closed in a changelist.

form

A screen displayed by certain Helix server commands. For example, you use the change form to enter comments about a particular changelist to verify the affected files.

forwarding replica

A replica server that can process read-only commands and deliver versioned files (depot files). One or more replicate servers can significantly improve performance by offloading some of the master server load. In many cases, a forwarding replica can become a disaster recovery server.

G

Git Fusion

A Perforce product that integrates Git with Helix, offering enterprise-ready Git repository management, and workflows that allow Git and Helix server users to collaborate on the same projects using their preferred tools.

graph depot

A depot of type graph that is used to store Git repos in the Helix server. See also Helix4Git and classic depot.

group

A feature in Helix server that makes it easier to manage permissions for multiple 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.

heartbeat

A process that allows one server to monitor another server, such as a standby server monitoring the master server (see the p4 heartbeat command).

Helix server

The Helix server depot and metadata; also, the program that manages the depot and metadata, also called Helix Core server.

Helix TeamHub

A Perforce management platform for code and artifact repository. TeamHub offers built-in support for Git, SVN, Mercurial, Maven, and more.

Helix4Git

Perforce solution for teams using Git. Helix4Git offers both speed and scalability and supports hybrid environments consisting of Git repositories and 'classic' Helix server depots.

hybrid workspace

A workspace that maps to files stored in a depot of the classic Perforce file revision model as well as to files stored in a repo of the graph model associated with git.

I

iconv

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, and propagate any outstanding changes from one set to another.

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. A job describes work to be done, such as a bug fix. Associating a job with a changelist records which changes fixed the bug.

job daemon

A program that checks the Helix server 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 form describing the fields and possible values for each job stored in the Helix server machine.

job view

A syntax used for searching Helix server jobs.

journal

A file containing a record of every change made to the Helix server's metadata since the time of the last checkpoint. This file grows as each Helix server transaction is logged. The file should be automatically truncated and renamed into a numbered journal when a checkpoint is taken.

journal rotation

The process of renaming the current journal to a numbered journal file.

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. A lazy copy points to the original versioned file (depot file). Lazy copies minimize the consumption of disk space by storing references to the original file instead of copies of the file.

license file

A file that 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 syntax for specifying a filename that is specific to an operating system.

lock

1. A file lock that prevents other clients from submitting the locked file. Files are unlocked with the 'p4 unlock' command or by submitting the changelist that contains the locked file. 2. A database lock that prevents another process from modifying the database db.* file.

log

Error output from the Helix server. 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. See also workspace view, branch view, and label view.

MDS checksum

The method used by Helix server to verify the integrity of versioned files (depot files).

merge

1. To create new files from existing files, preserving their ancestry (branching). 2. To propagate changes from one set of files to another. 3. 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 is stored in the Perforce database and is separate from the archive files that users submit.

modification time or modtime

The time a file was last changed.

MPM

Multi-Processing Module, 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

opened file

A file you have checked out in your client workspace as a result of a Helix Core server operation (such as an edit, add, delete, integrate). Opening a file from your operating system file browser is not tracked by Helix Core server.

owner

The Helix server user who created a particular client, branch, or label.

P

p4

1. The Helix Core server command line program. 2. 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.

P4PHP

The PHP interface to the Helix API, which enables you to write PHP code that interacts with a Helix server 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, Inc., a leading provider of enterprise-scale software solutions to technology developers and development operations (“DevOps”) teams requiring productivity, visibility, and scale during all phases of the development lifecycle.

project

In Helix Swarm, a group of Helix server 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.

proxy server

A Helix server that stores versioned files. A proxy server does not perform any commands. It serves versioned files to Helix server clients.

R

RCS format

Revision Control System format. Used for storing revisions of text files in versioned files (depot 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 but not make any changes.

remote depot

A depot located on another Helix server accessed by the current Helix server.

replica

A Helix server that contains a full or partial copy of metadata from a master Helix server. Replica servers are typically updated every second to stay synchronized with the master server.

repo

A graph depot contains one or more repos, and each repo contains files from Git users.

reresolve

The process of resolving a file after the file is resolved and before it is submitted.

resolve

The process you use to manage the differences between two revisions of a file, or two versions of a stream. You can choose to resolve file conflicts by selecting the source or target file to be submitted, by merging the contents of conflicting files, or by making additional changes. To resolve stream conflicts, you can choose to accept the public source, accept the checked out target, manually accept changes, or combine path fields of the public and checked out version while accepting all other changes made in the checked out version.

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 before a submit.

review access

A special protections level that includes read and list accesses and grants permission to run the p4 review command.

review daemon

A program that periodically checks the Helix server 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, typically designated with a pound sign (#).

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, a revision range, change numbers, label names, date/time specifications, or client names.

RPM

RPM Package Manager. 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 data

The combination of server metadata (the Helix server database) and the depot files (your organization's versioned source code and binary assets).

server root

The topmost directory in which p4d stores its metadata (db.* files) and all versioned files (depot files or source files). To specify the server root, set the P4ROOT environment variable or use the p4d -r flag.

service

In the Helix Core server, 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.

shelve

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. For the 'p4 status' command, by default the files opened and the files that need to be reconciled.

storage record

An entry within the db.storage table to track references to an archive file.

stream

A "branch" with built-in rules that determines what changes should be propagated and in what order they should be propagated.

stream depot

A depot used with streams and stream clients. Has structured branching, unlike the free-form branching of a "classic" depot. Uses the Perforce file revision model, not the graph model. See also classic depot and graph depot.

submit

To send a pending changelist into the Helix server depot for processing.

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 Helix server depot to a client workspace.

T

target file

The file that receives the changes from the donor file when you integrate changes between two codelines.

text file type

Helix server file type assigned to a file that contains only ASCII text, including Unicode text. See also binary file type.

theirs

The revision in the depot with which the client file (your 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.

trigger

A script that is automatically invoked by Helix server when various conditions are met. (See "Helix Core Server Administrator Guide" on "Triggers".)

two-way merge

The process of combining two file revisions. In a two-way merge, you can see differences between the files.

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

versioned file

Source files stored in the Helix server depot, including one or more revisions. Also known as an archive file. Versioned files typically use the naming convention 'filenamev' or '1.changelist.gz'.

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 your client workspace when you resolve a file. Also, the target file when you integrate a branched file.

License statements