HelixCore

# P4VS User Guide

2020.1

*March 2020*

PERFORCE

www.perforce.com

# Contents

# How to use this guide

This guide describes the installation, configuration, and operation of P4VS, the Helix Plugin for Visual Studio. With P4VS, developers who work with Microsoft Visual Studio have access to Helix Core features from within the Visual Studio interface. It is possible to check in and check out files, view the version history, and so on.

This section provides information on typographical conventions, feedback options, and additional documentation.

## Syntax conventions

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

| Notation | Meaning |
|---|---|
| `literal` | Must be used in the command exactly as shown. |
| *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.<br>▪ `p4 [g-opts] streamlog [ -l -L -t -m max ]` ***`stream1`*** `...`<br>means `1` or more stream arguments separated by a space<br>▪ See also the use on `...` in Command alias syntax in the *Helix Core P4 Command Reference*<br>**Tip**<br>`...` 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.

# What's new in this guide for this release

Following is a summary of new information with links to the most prominent topics. For a complete list, see the *P4VS Release Notes*.

- Added support for CodeLens with Microsoft Visual Studio 2019 version 16.1. If enabled, this feature lets you view in the editor the names of users who made changes to code and get a preview of up to five changelists. You can also open a changelist, the file history, or the Time-lapse View. For details, see "Finding code changes and references with CodeLens" on page 65.
- Added support for Helix Authentication Service. See the GitHub project at https://github.com/perforce/helix-authentication-service.

# Getting started with P4VS

This chapter provides an overview of P4VS, the Helix Plugin for Visual Studio, as well as instructions for installing and setting it up.



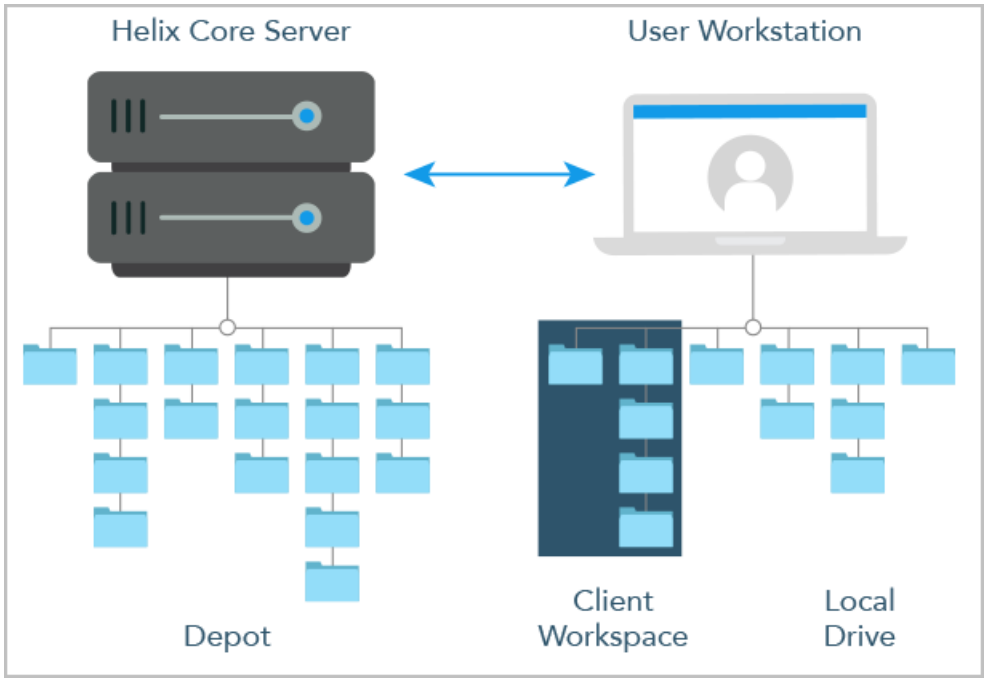
## About P4VS

P4VS, the Helix Plugin for Visual Studio , enables you to use Helix Core server, also referred to as Helix server, as your source control from within Visual Studio.

# Basic Helix Core server Terminology

- **Depot**: The shared repository where file revisions are stored and managed by Helix server.
- **Workspace**: The area on your computer where you work with your copies of files that are under Helix server control.
- **Helix Core server**: Helix server, the application that manages file revisions.
- **Changelist**: A group of files, with a description explaining how you have changed them (for example, `Fix bug #123`). Changelists are assigned numbers by Helix server so you can track them. Changelists enable you to group related files and submit them together.

The following diagram shows the relationship between workspace and depot:




# Basic Tasks

- **Get revision**: Retrieve a copy of a file version from the depot. Helix server also uses the term *sync* to mean *get revision*.
- **Check out**: Enables you to change the file.
- **Mark** for add or delete: indicates that the file is added to or deleted from the depot when the changelist is submitted.
- **Revert** a file: Discard any changes you have made to an open file. If you open a file for edit and make changes, then change your mind and revert the file, Helix server reloads the last version you got from the depot and discards your changes.

- **Submit** a changelist: Update the depot to reflect any changes you have made to files in the changelist. Submitting is an all-or-nothing operation: if there is a problem submitting one file in a changelist, none of the other files in the changelist are updated.

## Using Solution Explorer with P4VS

Solution Explorer provides access to most P4VS functionality and status information.

- When you right-click a file in Solution Explorer, all P4VS actions enabled for that file are available for selection in the context menu.
- Badges on file icons indicate Helix server status:

| | |
|---|---|
|  | Marked for add |
|  | Marked for delete |
|  | Checked out by you |
|  | Checked out by another user |
|  | Locked |
|  | Version in workspace is not latest version |
|  | Workspace version is up to date |
|  | Needs resolve (conflicting changes have been made) |
|  | File not in Helix server depot |
|  | Marked for integrate |
|  | Ignored |
|  | File is lazily loaded, whenever operated on by P4VS |

Right-click a file and select **Refresh** in the context menu to refresh Helix server status information for the file and any of its children.

> **Note**
> If you select **Automatically update files status when selection changes** in the P4VS Preferences, the file's Helix server status updates automatically when you click or hover over the file icon, without having to click **Refresh**.

## Using P4VS toolbars in Visual Studio

P4VS provides the following toolbars that you can use with Visual Studio:

- The **Helix Connection** toolbar displays your current Helix server service connection (hostname:port, workspace, and user), as well as the pending changelist you are working in. If you are not connected to Helix server, the Connection list shows **OFFLINE**.
- The **Helix Views** toolbar provides access to workspaces, file history, jobs, submitted and pending changelists, labels, and streams.
- The **P4VS** toolbar provides the same menu of P4VS options as the Solution Explorer context menu.

To enable these toolbars, select **View > Toolbars**.

To enable these toolbars and customize them to show only a subset of the available options, select **Tools > Customize**.

## *For more information*

Watch our P4VS tutorial video: https://www.perforce.com/video-tutorials/plugin-visual-studio

For more information about how to use Helix server, see *Solutions Overview: Helix Version Control System*.

For the P4VS release notes, see https://www.perforce.com/perforce/doc.current/user/p4vsnotes.txt.

# Installing P4VS and enabling the extension in Visual Studio

To use P4VS with Microsoft Visual Studio, you must install the plugin and then enable it within Visual Studio.

## *Installing P4VS in Visual Studio*

1. Make sure Microsoft Visual Studio is closed.
2. Download the P4VS Visual Studio Extension Installer file, **`p4vs.vsix`**.
3. Open the installer file.
4. Select the version of Visual Studio that you want the extension to install to.
5. Click **Install**.
6. The installer screen displays an **Installation Complete** message.

## *Enabling P4VS in Visual Studio*

> **Note**
> This step is not needed when you install a P4VS upgrade.

1. Open Visual Studio.
2. In Visual Studio, go to **Tools > Options**.
3. Select **Source Control > Plug-in Selection**.
4. In the **Current source control plug-in** drop-down list, select **P4VS - Helix Plugin for Visual Studio**.

# Setting P4VS preferences

To set P4VS preferences in Visual Studio, go to **Tools > Options**. You can specify preferences on the following nodes in the **Options** dialog under **Source Control**:

- "Helix Core - Connections" below
- "Helix Core - Data Retrieval" on the next page
- "Helix Core - Diff/Merge/Reviews" on page 17
- "Helix Core - General" on page 17
- "Helix Core - Ignoring Files" on page 19
- "Helix Core - Logging" on page 20

In addition, you can:

- Specify keyboard shortcuts for P4VS commands. Go to **Tools > Options > Environment > Keyboard**.
- Turn on CodeLens functionality for P4VS. Go to **Tools > Options > Text Editor > All Languages > CodeLens** and select **Show Helix Commit Provider**.

## *Helix Core - Connections*

Set the following preferences to determine how you connect to Helix Core server in Visual Studio. To save your changes, click **OK**.

### When opening a project under source control

- **Show the Helix Core server Connection**: Prompt for connection settings whenever you open a project that is under Helix server source control.
- **Connect to the server using my most recent settings**: Without prompting for connection settings, reconnect to the Helix server you were connected to during your last session.
- **Connect to the server using solution-specific settings**: Without prompting for connection settings, connect to the Helix server you last used for the solution or project that you are opening.
- **Connect to the server using my Helix Core environment settings**: Connect using Windows environment variables for Helix server connections, which you set using the Helix server

Command-Line Client or P4V, the Helix Visual Client. For more information, see the *Helix Core Server User Guide* or the P4V help.

### 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.

### Saved connections

If you have saved connections, they remain in the list of 5 most recently used connections. If you no longer need to use a saved connection, select it in this list and click **Remove** to delete it from the preferences.

## *Helix Core - Data Retrieval*

Set the following preferences to determine how P4VS retrieves data from Helix server:

### Data retrieval:

- **Check server for updates every**: Specifies how often P4VS checks Helix server for updated file information. Frequent checks enable P4VS to display current file status but increase the workload on Helix server.
- **Maximum number of files displayed per changelist**: Specifies the maximum number of files displayed in a changelist, to help minimize the time required to handle and submit very large changelists. This setting affects only the display of changelists, and does not limit the number of files that a changelist can contain.
- **Maximum size of files to preview**: Limits the size of image files displayed in the **Preview** tab, to limit the amount of image data sent from Helix server to P4VS.
- **Number of changelists, jobs, or labels to fetch at a time**: Specifies the number of specifications read in each batch fetched, to minimize server load and maximize P4VS performance. To retrieve all entries, specify `0`.
- **Automatically update file status when selection changes**: Select to enable the Helix server status badges in Solution Explorer to update automatically when you click or hover over the file icon, without having to click **Refresh**. Deselect to improve performance.

### Optimize file status retrieval:

> **Note**
> The **Treat Solution/Projects as directories when selected**, **Preload file state**, and **Lazy load file**

**state** options are used to tune the performance of P4VS for your environment. If none of these options are selected, P4VS will load the metadata for each file individually. Unless you have a small project you should look to use one of these options.

- **Optimize file state retrieval**: Select to apply optimizations on retrieving file state. Subordinate optimization options include:
  - **Treat Solution/Projects as directories when selected**: Select to treat solutions and projects as directories when P4VS runs Helix server commands.

    Use this option to improve performance when working with solutions that contain a large number of projects or files. Do not use this option if the directories in the solution contain a large number of other Helix server controlled files that are not included in the solution.

    This option does not require that all the files and directories referenced by the solution are under the solution directory.
  - **Preload file state**: Select to preload the metadata for all of the files in the Helix server depot in or under the directory containing the solution file.

    Use this option to improve performance when loading small to medium sized solutions where all the files that make up the solution are under the solution root. Only use this when there are few if any files under the solution root that are Helix server controlled that are not part of the solution.

    This option tends to work best with solutions and projects created and managed by Visual Studio. Do not use this option if the files for the solution are intermixed in directories with large numbers of other Helix server controlled files. This option will provide little improvement in performance if most of the files composing the solution are outside of the solution root.
- **Lazy load file state**: Select to only load the Helix server metadata files in the solution as they are operated on using P4VS.

  With this option, Helix server metadata for a file is only retreived from the server when you select a P4VS operation on a file. At that point, the Helix server metadata will be obtained from the server, the operation performed, and the Helix server metadata updated to reflect the results of the operation. When this option is selected, the file will by badged in the solution explorer to indicate that its status is unknown. After a Helix server operation is performed on a file, it will be badged to indicate its current state.

  Use this option to improve performance loading large to very large sized solutions where you are interested in only working on a few select files. This is best for solutions and projects which include large amounts of code from libraries or frameworks, large numbers of asset files such as graphical elements for a game, or large numbers of files that are generated by another development system or plugin. This option is also useful in situations where the connection to the Helix server server is over a slow network or VPN.
  - **Full menu**: Select to allow the full P4VS menu to be displayed on files that have not had their metadata loaded.

This option is displayed if the **Lazy load file state** option is enabled. This allows you to perform an operation on a file before P4VS loads its state. If this option is not selected, you are given the option to refresh the file which will load the metadata of the file from the server and then enable the appropriate P4VS operations on that file. Please note that if you choose this option and perform P4VS operations on a file that are not valid based on its current state, you are very likely to get error messages back from those operations.

- **Do not optimize**: Select this to disable all optimizations related to file state retrieval.

## Helix Core - Diff/Merge/Reviews

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

- **P4Merge**: The Helix Core companion diff tool.
- **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. Helix server replaces these placeholders with the actual filenames when calling the diff application.

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

- **P4Merge**: The Helix Core companion merge tool.
- **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`

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

To have Helix Swarm reviews open in an external browser (instead of in a new tab in Microsoft Visual Studio, which is the default):

- Select the **Open Swarm reviews in external browser** check box.

## Helix Core - General

Set the following display and file behavior preferences:

## Display:

- **Use OS format for dates**: Use the date format that the operating system uses.
- **Format dates using Helix standard (yyyy/mm/dd hh:mm:ss)**: Use the Helix server format.

## Files and folders:

- **Warn before reverting files**: If selected, P4VS displays a prompt before reverting files.
- **Lock files on checkout**: If selected, P4VS locks files every time you check them out. Locks prevent other users from checking in changes while you work on a file.
- **Prompt for changelist when checking out, adding, or deleting files**: If selected, P4VS prompts you to select a changelist when files are about to be checked out, added, or deleted in Helix Core server. If cleared, Helix server performs the action but does not prompt you to select a changelist. Instead, it uses the active changelist, which is either the default or the changelist specified in the Connection toolbar.
- **Mark for delete in Helix Core server when deleting files:** If selected, files that you delete in Visual Studio are marked for delete in Helix Core server.
- **Check out writable files on save**: If selected, P4VS automatically checks out files that are writable in the workspace.
- **Automatically add new files to Helix Core server**: If selected, P4VSmarks new files for add in a pending changelist.

  This option works together with the **Prompt for changelist when checking out, adding, or deleting files** option to determine prompting behavior when you add new files to a project or solution that is under Helix server control:

  - If both this option and the **Prompt for changelist** option are enabled, P4VS prompts you to mark new files for add.
  - If this option is enabled and the **Prompt for changelist** option is disabled, new files are automatically added to the default changelist without any prompt.
  - If this option is disabled and the **Prompt for changelist** option is enabled, no prompts will appear; you must manually mark new files for add.
- **Update related projects when reverting moved files**: If selected, P4VS reverts file renames or moves in Visual Studio when you revert a Helix server rename/move operation.

  If you do not select this option, Visual Studio continues to show the new file name or location despite the fact that Helix server has reverted the file to its original name or location.

  > **Note**
  > If you revert a folder rename/move in P4VS, you must manually revert the name or location in Visual Studio, regardless of your preference selection.

- **Use Visual Studio to view file versions**: If selected, P4VS shows previous revisions of a file

(from the **File History** dialog, for example) in a Visual Studio editor window.

You can use this option to view file revisions or shelved files the same way that you would view an editable file in the Visual Studio IDE.

- **When starting to edit an out of date file:** If **Always ask to sync the file** is selected (default), P4VS prompts you before syncing to prevent you from losing any work. If the check box is cleared, P4VS always syncs the file, without prompting.

### Project and solution files:

- **Tag project and solution files as controlled by P4VS**: If selected, P4VS writes tags to the solution and project files that are under Helix server control. The default is not to tag project and solution files; P4VS does not need to tag these files to know that they are under Helix server control.
- **Set the location of new projects to the current workspace root directory**: If selected, P4VS puts new projects in the current workspace root directory. The **Location** field in the **New Project** dialog will be populated by default with the current workspace directory.
- **Warn if solution is outside workspace root:** If selected, P4VS displays a warning message when a solution is not in the client map. If you do not store solutions in source control or you do not want to be alerted if a solution is created outside the workspace root, you can clear this check box.

## *Helix Core - Ignoring Files*

To avoid adding and checking in files that do not belong in the repository, you can exclude individual files or file types (for example, build or release artifacts) from source control using **Ignore Lists**. For more information, see "Excluding Files from Helix server Control" on page 46.

You can set the following preferences:

- **Enable Ignore Lists for specifying files to ignore when marking for add**: Select to enable P4VS to use **Ignore Lists** to keep individual files or file types from being added to the Helix server repository.

  If you have already set an Ignore List file as the `P4IGNORE` environment variable on your local machine, that file name appears by default in the **Name** field. If not, enter a file name or accept the standard default, `.p4ignore.txt`. The first time you select **Edit Ignore List** or **Add to Ignore List** for a file in a folder in Solution Explorer, P4VS adds an **Ignore List** file with this file name to that folder.

  > **Note**
  > Your local `P4IGNORE` environment variable will be updated with the file name that you enter here. If other Helix server clients (such as `p4` or P4V) on your local machine use **Ignore List** files, be sure to use the same file name as you use with those clients.

- **Automatically add new Ignore Lists to solution or project**: Select to have new **Ignore Lists** appear in the Solution Explorer.

  If you do not select this option, the **Ignore List** file will be hidden in Solution Explorer.
- **Prompt when creating a new Ignore List**: Select to have P4VS prompt you when you select **Add to Ignore List** in Solution Explorer to add an **Ignore List** to a folder that does not yet have one.

  If you do not select this option, the system creates the **Ignore List** without prompting.
- **Automatically ignore new Ignore Lists (add Ignore Lists to themselves)**: Select to have P4VS automatically add the **Ignore List** file to itself to prevent the **Ignore List** from being added to the Helix server depot.
- **Automatically add new Ignore Lists to Helix Core server**: Select to have P4VS automatically add new **Ignore Lists** to the Helix server depot.

  This option works only when the **Automatically ignore new Ignore Lists (add Ignore Lists to themselves)** option is not selected. If that option is not selected, and you do not select the **Automatically add new Ignore Lists to Helix Core** option, then you must manually mark the **Ignore List** file for add to add it to the repository.

## Helix Core - Logging

Set the following logging preferences. You can view P4VS log messages in the **Output** window in Visual Studio if you select **Helix Core Source Control** in the **Show output from** drop-down list in the **Output** window.

- **Show p4 reporting commands**: Specifies whether the Output window in Visual Studio displays all commands issued by P4VS, including commands issued by P4VS to obtain status information from Helix server.
- **Show p4 command output for file operations**: For verbose log messages, enable this option.
- **Enable logging to file**: Logs P4VS activity to the specified file.
  - **Name**: Specifies the name and location of the log file.
  - **Size**: Specifies the maximum size of the log file.

## Keyboard shortcuts

Go to **Tools > Options > Environment > Keyboard.** You can find P4VS commands by entering **P4VS** in the **Show commands containing** field.

For more information about creating keyboard shortcuts in Visual Studio, see the Microsoft Visual Studio help.

# Connecting to a Helix server

Connections enable you to access Helix server to submit and obtain access to files under Helix server control. You use the **Open Connection** dialog both to define connections and to open them in P4VS.

You can:

- Define a new connection
- Define a connection using environment variables
- Open a defined connection

## *Defining a new Helix server connection*

To define a new connection to a Helix server in Visual Studio:

1. Open the **Open Connection** dialog. You can open this dialog the following ways:
   - Add a new project in Visual Studio in the **New Project** dialog. In this dialog, select the **Add to source control using P4VS - Helix Plugin for Visual Studio** check box; then click **OK**.

     The **Open Connection** dialog appears unless:
     - You did not select the check box.
     - You have set the connection settings in **Tools > Options > Source Control > Helix Core - Connections** to default to an option other than **Show the Helix Core server Connection**. In this case, P4VS attempts to connect without showing the **Open Connection** dialog.
     - You are already connected. In this case, P4VS does not attempt to connect and marks the new project for add.
   - Open a solution or project under Helix server source control in Visual Studio.

     The **Open Connection** dialog appears unless you've set the connection settings in **Tools > Options > Source Control** to default to the last Helix server connection or to connection settings defined in your environment variables.
   - Go to **File > Open Connection to a Helix Core server** in Visual Studio.
2. **Enter the server name and port number for this connection using `host:port_number`.**

   If your Helix server is enabled for SSL (Secure Sockets Layer) encryption, use the following syntax: `ssl: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 the **New…** button 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 the **New…** button.
   - In the **New Workspace** dialog, enter a workspace name and click **OK**.
   - In the **Workspace** dialog, entered the required information.

   For more information on setting up client workspaces, see "Managing workspace specifications" on page 25.
5. **Click** OK**.**

   P4VS connects to the specified Helix server. If the user you are connecting with does not have a p4 ticket, P4V prompts you for your password.

> **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.

## *Setting Helix server connection settings using environment variables*

You can set Windows environment variables for Helix server connection settings, which makes the settings available to P4VS and other Helix Core client applications (for example, P4EXP, the Helix Plugin for File Explorer). Set Helix server connection settings as environment variables and configure your connection preferences in **Tools > Options > Source Control** to default to the environment variables.

Another approach is to create a configuration file that stores your Helix server environment variables. You can then point to the configuration file using the environment variable `P4CONFIG`. P4VS searches the current working directory and its parents for the file. If the file exists, then P4VS uses the variable settings within the file. `P4CONFIG` makes it easy to switch Helix server settings when switching between different solutions or projects. If you place a configuration file in each of your client workspaces and set `P4CONFIG` to point to that file, your Helix server settings change to the settings in the configuration files automatically as you move from directories in one workspace to another.

For more information about how to use `P4CONFIG` with P4VS, see "Setting Helix server environment variables using P4CONFIG" below.

For more information about Helix server environment variables, see the *Helix Core P4 Command Reference* and "Setting P4VS preferences" on page 14.

## *Opening a defined Helix server connection*

To open a Helix server connection that you have already used, select the connection from the drop-down list in the **Open Connection** dialog. You can also configure P4VS to automatically open the connection that you used most recently when you open a project. See "Setting P4VS preferences" on page 14.

## Setting Helix server environment variables using P4CONFIG

`P4CONFIG` is an environment variable that you can use to point to a file that stores other Helix server environment variables. The current working directory and its parents are searched for the file. If the file exists, then the variable settings within the file are used.

`P4CONFIG` makes it easy to switch Helix server settings when switching between different solutions or projects. If you place a configuration file in each of your client workspaces and set `P4CONFIG` to point to that file, your Helix server settings change to the settings in the configuration files automatically as you move from directories in one workspace to another.

To use `P4CONFIG` to switch settings between client workspaces on P4VS, you must create separate Visual Studio shortcuts for each workspace, setting the `Start in` property as the workspace directory (which is also the directory where the configuration file resides). If you launch Visual Studio using a shortcut defined this way, P4VS will read the Helix server settings from the configuration file in that workspace's directory. This is required because Visual Studio otherwise uses its own directory as the current working directory.

To use `P4CONFIG` with P4VS:

1. Create a configuration file that contains the Helix server environment variable settings you want, and put it in the workspace directory for the relevant Visual Studio solution.
2. Using `p4`, the Helix server Command Line Client, unset the `P4CLIENT`, `P4PORT`, and `P4USER` environment variables and set `P4CONFIG` to the configuration file name.
3. Using P4VS, go to **Tools > Options > Source Control > Helix Core - Connections** and select **Connect to the server using my Helix Core environment settings**.
4. Create a Windows desktop shortcut for Visual Studio that is dedicated to the workspace with which you want to use the configuration file.
   a. Right-click on the desktop and select **New > Shortcut**.
   b. Enter the location of the Visual Studio executable and click **Next**.
   c. Enter a shortcut name and click **Finish**.
   d. In the shortcut properties, under **Start in**, enter the workspace directory where the configuration file is located, and click **OK**.

   Repeat for each workspace for which you want a different configuration file.
5. Whenever you want to work in that workspace using the configuration file settings, use the shortcut to launch Visual Studio.

For more information about `P4CONFIG` and Helix server environment variables, see the *Helix Core P4 Command Reference* and "Setting P4VS preferences" on page 14.

## Customizing context menus

You can use Visual Studio customization functionality to add or remove P4VS commands in context menus.

To add or remove a P4VS command:

1. Go to **Tools > Customize** and open the **Commands** tab.
2. In the **Menu bar** drop-down, select the menu you want to customize.

   The P4VS menus begin with **File | Helix**.
3. Under **Controls**, select a command to delete or move, or select **Add Command** to select a command to add to the menu.

   Many of the P4VS commands are under the **File** and **View** categories.

   > **Note**
   > There are many P4VS commands with names that are similar to native Visual Studio or other plug-in commands. If you have any questions about which commands belong to P4VS, contact your Helix server administrator.

For more information, see the Microsoft Visual Studio help.

# Managing workspace specifications

A workspace specification defines which portion of the depot can be accessed from that workspace and specifies where local copies of files in the depot are stored. This location is called the workspace. A computer can contain multiple workspaces. A workspace is required when connecting to Helix server if you intend to work with files.

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. (In fact, you cannot manually edit the view of a stream workspace.) If you use classic depots, you must define and maintain the workspace view manually.



## *Creating workspaces*

To create a new workspace in P4VS:

1. Open the **Open Connection** dialog.

   For more information, see "Connecting to a Helix server" on page 21.
2. Click the **New** button next to the **Workspace** field to open the **New Workspace** dialog.
3. Enter a workspace name and click **OK**.
4. In the **Workspace** dialog, view or enter the following settings:

| Setting | Description |
|---|---|
| `Workspace` | Workspace name. Defaults from the **New Workspace** dialog. |
| `Owner` | The user who created the specification. Defaults to you when you create a new workspace. |
| `Host` | (optional) The computer where the workspace resides. To enable the workspace to be used from any machine, leave this field blank. |
| `Submit options` | Configures what happens when users submit files. |

| Setting | Description |
|---|---|
| `Line endings` | The line-end convention used for storing text files on the workspace computer:<br>■ Local: Uses the workspace platform default<br>■ Unix: `LF`<br>■ Mac: `CR`<br>■ Windows: `CRLF`<br>■ 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) |
| `Description` | Your own explanation of the purpose of the workspace, or any related information you want to specify. |
| `Root` | Workspace root directory where you want local copies of depot files stored. |
| `Alt Roots` | For workspace specifications used from hosts on different platforms, a list of workspace roots in host-platform-specific syntax. |
| `Options` | ■ `allwrite`: All files in the workspace are writable (can be modified).<br>■ `clobber`: Syncing files overwrites writable files on the workspace.<br>■ `compress`: Compresses data sent between the workspace and Helix server.<br>■ `locked`: Only the owner of the workspace can use, change, or delete the workspace specification.<br>■ `modtime`: Modification time for files edited in the client workspace is set to the time when the file is submitted to the depot.<br>■ `rmdir`: Deletes a workspace folder if all the files contained in the folder are removed. |
| `Stream Root` | Root directory for a workspace associated with a mainline stream. For more information on streams and how Helix server handles stream workspaces, see "Stream workspaces" on the next page. |

| Setting | Description |
|---|---|
| `View` | The workspace view determines which portions of the depot are visible in your **Workspace Tree** and where local copies of depot files are stored in your workspace. If you use streams, the workspace view is generated and updated automatically. For more information on workspace views, see "Defining a workspace view" below. |

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

## Changing your workspace

To change the workspace you are using, use the **Open Connection** dialog and specify the workspace in the **Workspace** field.

For more information, see "Connecting to a Helix server" on page 21.

## Viewing workspaces

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

- Go to **View > Workspaces** in the Visual Studio menu bar to open the **Workspaces** tool window.

  Click a workspace row to display the details of the client workspace specification.

  To change the order in which columns are displayed, drag the column headings right or left to the desired position. To sort by column, click the sort arrow on a column heading.

- Open the **Open Connection** dialog and click the **Workspace Browse…** button to open the **Workspace Browser** dialog.

  Click a workspace row to display the details of the client workspace specification.

  For more information, see "Connecting to a Helix server" on page 21.

## Stream workspaces

If you work with streams, P4VS uses workspaces differently than it does with classic depots. For more information, see the Streams chapter in the *Helix Core Server User Guide*>.

## Defining a workspace view

The workspace view (sometimes called a *client* view) determines which portions of the depot are available for you to work with in P4VS and where local copies of depot files are stored in your workspace. If you use streams, the workspace view is generated and updated automatically. If you use classic depots, you must maintain the view manually, as described in this topic.

To define or change the workspace view for an existing workspace:

1. Select **View > Workspaces**. The **Workspaces** tab is displayed.
2. Right-click the workspace and select **Edit Workspace**. The **Workspace** form is displayed.
3. Edit the **View** field. Define the view as described under Syntactic view specification.
4. When you have finished editing, save your changes.

To define the workspace view for a new workspace:

1. Open the **Open Connection** dialog.

   For more information, see "Connecting to a Helix server" on page 21.
2. Click the **New** button next to the **Workspace** field to open the **New Workspace** dialog.
3. Enter a workspace name and click **OK**.
4. In the **Workspace** dialog, edit the **View** field. Define the view as described under Syntactic view specification.

## Syntactic view specification

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

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

For details about client view syntax, refer to the *Helix Core Server User Guide*.

# Managing files

This chapter discusses how to manage files using P4VS.



## Putting a project or solution under Helix server source control

P4VS requires that your work be included in a project or solution file.

> **Note**
> Make sure that your project or solution and all files included in it reside in the workspace (your client

directory) being used by your Helix server connection.

The way to put the project or solution under Helix server source control depends on your configuration:

- If P4VS is the active source control provider but is not set to automatically add new files to Helix server, select the **Add to source control using P4VS** check box in the **New Project** dialog when you create the solution. This causes P4VS to mark the files for add. You then only need to submit the pending changelist. For details, see "Option 1: Existing project or solution with P4VS as active source control provider" below.
- If P4VS is the active source control provider and set to automatically add new files to Helix server, P4VS automatically marks the files for add when you create the solution, regardless of whether the **Add to source control using P4VS** check box in the **New Project** dialog is selected. You then only need to submit the pending changelist. For details, see "Option 2: New project or solution with P4VS as active source control provider" below.
- If P4VS is not the active source control provider, use the **Publish** option (Visual Studio 2015) or the **Add to Source Control** option (Visual Studio 2017) in the status bar, available after you create the solution. Note that this option is not available in Visual Studio 2013.

  For details, see "Option 3: New project or solution without P4VS as active source control provider" on the facing page.

For more configuration information, see "Setting P4VS preferences" on page 14.

## *Option 1: Existing project or solution with P4VS as active source control provider*

1. In the **Solution Explorer**, select the project or solution that should be placed under source control.
2. Follow the procedure described in "Adding files to the depot" on the facing page.

## *Option 2: New project or solution with P4VS as active source control provider*

1. In the **New Project** dialog, if P4VS is not set to automatically put files under Helix server source control, select the **Add to source control using P4VS** check box under the **Browse** button.
2. Click **OK**.

   If you are offline, the **Open Connection** dialog opens; continue with step 3. Otherwise, continue with step 4.
3. In the **Open Connection** dialog, enter your Helix server connection settings and click **OK**.

P4VS opens the project or solution and any related files for add.

4. Continue with submitting the changelist.

## *Option 3: New project or solution without P4VS as active source control provider*

1. In the status bar, at the bottom right of the window, click **Publish** (Visual Studio 2015) or **Add to Source Control** (Visual Studio 2017), and then select **P4VS - Helix Plugin for Visual Studio**.

   Note that this option is not available in Visual Studio 2013.

2. In the **Open Connection** dialog, enter your Helix server connection settings and click **OK**.

   The files in the **Solution Explorer** now display a red plus sign to indicate that they are marked for add.

   The **Add to Source Control** option in the bottom right of the window changes to **1 Pending Change** or ***X* Pending Changes** (if you have other pending changelists in addition to the default changelist).

3. Continue with submitting the changelist.

# 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.
2. Submit the changelist, which copies the file to the depot.

**To open a file for add:**

1. In the Solution Explorer, browse to the file you want to add.

   If a file does not reside in the depot, its icon is marked with a blue question mark.

2. Right-click the file and select **Mark for Add**.

   A P4VS dialog opens asking you to add the files to the Helix server depot.

3. Select the pending changelist you want to use for submitting the file.
4. Click **OK**.

   The file icon in Solution Explorer displays a red plus sign indicating that it is open for add.

**To submit the changelist:**

1. In the Solution Explorer, right-click the file and select **Submit**.
2. In the **Submit Files** dialog, enter a description of the change and click **Submit**.

   The new file is added to the depot.

> **Note**
> If you add a file to a solution that is already under Helix server control, you are prompted to put the new file under Helix server control. If you enabled the **Automatically add new files to Helix Core** option and disabled the **Prompt for changelist when checking out or adding files** option under **Tools > Options > Source Control**, the file is marked for add and placed in a changelist without any prompting. For more information about these options, see "Setting P4VS preferences" on page 14.

For more information, see "Checking in files and working with changelists" on page 34.

## Opening a project or solution in the Helix server depot

To open a project or solution that has been checked into a Helix Core depot:

1. Go to **File > Helix > Open Solution/Project in Helix Core server**.
2. In the **Choose Solution/Project in Depot** dialog, expand the tree to find the solution or project you want to open.

   Select **Filter by client workspace** to limit the depot tree to the solution and project files that are included in the current workspace view.

   If you cannot expand and view the contents of the depot tree, you are not connected to Helix server. Click **Open Connection** to connect.
3. Click the file and click **OK** to open it in Visual Studio.

## 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 Solution Explorer, open the folder containing the file you want to retrieve. The icons indicate the status of the files; see "Getting started with P4VS" on page 10 for details.

To get the latest revision:

1. Right-click the file or folder in the Solution Explorer.
2. Select **Revisions > Get Latest Revision**.

To get a previous revision:

1. Right-click the file in the Solution Explorer and select **Revisions > Get Revision**.
2. In the **Get Revision** dialog, specify the revision you want:
   - Under **Get or replace the following files/folders**, you can select specific files or folders to retrieve.

- To to specify a revision by changelist number, label, workspace, or date, choose the method from the **Specify revision using**: drop-down list and specify the value in the edit field.
- Select **Force Operation** to retrieve the selected revision into your workspace even if the workspace already has the file. This option does not affect open files.
- Select **Only get revisions for files listed in changelists** to retrieve only those files that are included in changelists.
- If you are specifying a revision by label, you can ensure that your workspace contains only the labeled file revisions by selecting **Remove files from workspace if they are not in label.**

3. Click **Get Revision** to retrieve the files to your workspace.

## Checking out and editing files

Before you edit a file, you must check it out of the Helix server depot.

To check out and edit a file:

1. In the Solution Explorer, find the file that you want to edit.

   If necessary, retrieve the correct revision to your workspace. For more information, see "Retrieving files from the depot" on the previous page.

2. Right-click the file and choose one of the following:
   - **Checkout *filename*** to check out only the selected file
   - **Checkout All in Project** to check out the project file and all files in the project
   - **Checkout All in Solution** to check out the solution file and all files in the solution

   When you check a file out, it is placed in a changelist.

3. Make your changes.
4. To check your revised version back into the depot so that other users can view your changes and edit 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. For more information, see "Checking in files" on the next page.

To display a file without checking it out, double-click the file icon. It opens in read-only mode.

To lock a file to prevent others from checking it out while you are working on it, right-click the file icon and select **Manage Files > Lock**. To unlock it, right-click and select **Manage Files > Unlock**.

> **Note**
> When you try to edit or save a file that is checked into Helix server, P4VS asks if you want to check it out (and save it, if you are attempting a save). It also gives you the following options:

- **Don't show this dialog again (always use the default changelist)**: always check out (when opening for edit) or check out and save (when saving edits) and add to the default changelist, without prompting from P4VS.
- **Do this for all files being saved** (or **edited**): if your save or edit operation involves multiple files, select this option to check out (when opening for edit) or check out and save (when saving) all files in the current operation without having the P4VS dialog prompt you for each file individually.

# Checking in files and working with changelists

To check in a file, you must submit a *changelist*. Whenever you mark files for add or delete, check them out, integrate (merge or copy), 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.



## *Checking in files*

To check in files (submit a changelist):

1. Open the **Submit** dialog by doing one of the following:
   - Right-click the icon of a file that is checked out, marked for add, or marked for delete, and choose **Submit…** to open the **Submit Files** dialog.
   - Go to **View > Pending changelists** or, in the status bar, click **1 Pending Change** or **<number of changes> Pending Changes** to open the **Pending** dialog. Then right-click a changelist and choose **Submit…** to open the **Submit Files** dialog.

   Note that the **Submit Changelist** and **Submit Files** dialogs are functionally identical; they differ only in how you access them.
2. In the **Submit Changelist** or **Submit Files** dialog, enter a description or edit the existing description, and select the files you want to check in.

   You can also perform the following actions:

- Remove files from the changelist.
- Revert unchanged files in the changelist (removing the unchanged files from the changelist, canceling the check-out, and leaving them synced to the version you originally checked out) or submit only changed files (moving the unchanged files to the default changelist after the current changelist is submitted).
- Check out submitted files after you submit them.
- Associate the changelist with a job and set the job status upon submit. For more information about jobs, see "Using jobs (defect tracking)" on page 67.
- Perform a diff on a file pending submission by right-clicking the file and selecting **Diff Against Have Revision**. For more information, see "Comparing files using diff" on page 48.

3. Optional) Click **Save** to save your changelist options without checking in files.
4. Click **Submit** to check in your files.

## Displaying changelists

P4VS lets you view both pending and submitted changelists. You can also edit some of the information in changelists that have already been submitted.

**To open changelists from the View menu:**

1. Go to **View > Pending changelists** or **View > Submitted changelists** to open the **Pending** or **Submitted** tool windows.

   To change the order in which columns are displayed, drag the column headings right or left to the desired position. To sort by column, click the sort arrow on a column heading.

2. (Optional) Filter the displayed changelists:

   Enter your filter criteria in the **Folder/file**, **User**, and **Workspace** fields.

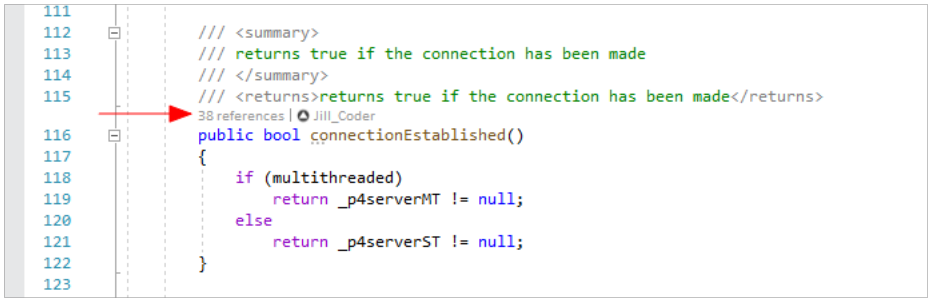
   To filter by file, enter the full path of the file in the workspace. The filtering process is case-sensitive.

   Click **Filter**.

3. View changelist details by doing one of the following:
   - Submitted changelists only: Select a changelist to display details in the fields below the changelist viewer, including description, files, jobs, and user.
   - Click the arrow next to the changelist row to expand the changelist row and view the files included in the changelist.
4. For edit options, continue with "Editing changelists" on the next page.

**To open a submitted changelist from within a file:**

1. Above a class or function, click the gray user name to display a list of changes, as indicated in the following figure.




2. Double-click the change for which you want to see the changelist.

   The **Submitted Changelist** dialog opens.

   For more information on this feature, see "Finding code changes and references with CodeLens" on page 65.

# Editing changelists

P4VS lets you edit both pending and submitted changelists. You can edit and perform actions on a pending changelist using the **Pending** tool window and the **Pending Changelist** dialog. You can also edit details in changelists that have already been submitted using the **Submitted** tool window and the **Submitted Changelist** dialog.

**To work with changelists from the Pending tool window:**

1. Go to **View > Pending Changelists** to open the **Pending** window. Alternatively, in the status bar, click **1 Pending Change** or ***X* Pending Changes**.
2. Right-click the changelist, a file, or an associated job and select any of the following actions:

> **Note**
> These menu items are dynamic. Some only appear for numbered changelists or when you have Helix Swarm installed.

| Item to right-click | Action | More information |
|---|---|---|
| Pending changelist | Submit | Opens the **Submit Files** dialog. See "Checking in files" on page 34. |
| | Revert Unchanged Files, or Revert Files | Reverts all files in the changelist that have not been modified, or all files in the changelist, regardless of whether they have been modified or not. See "Checking in files" on page 34. |
| | Move All Files to Another Changelist | Opens a dialog that lets you select a different changelist. See "Moving a file to another changelist" on page 41. |
| | Shelve files | Opens the **Shelve** dialog. See "Shelving files" on page 51. |
| | Submit Shelved Files | Silently submits the files. See "Submitting shelved files" on page 53. |
| | Unshelve Files | Opens the **Unshelve** dialog. See "Unshelving files" on page 53. |
| | Delete Shelved Files | Deletes the shelved copies of files. P4VS prompts you for confirmation. |
| | Diff Files Against Have Revisions | Opens both file revisions in P4Merge. See "Comparing files using diff" on page 48. |
| | New Pending Changelist | Opens the **Pending Changelist** dialog. |
| | Edit Pending Changelist | Opens the **Pending Changelist** dialog. |
| | Request New Swarm Review or Update a Swarm Review | See "Working with reviews in Swarm" on page 70. |
| | Change Ownership | Opens a dialog that lets you set the user and workspace. |

| Item to right-click | Action | More information |
|---|---|---|
| | Refresh Pending Changelist List, or Refresh Pending Changelist | Updates the **Pending** tool window or the pending changelist that you have selected. |
| File or files in a pending changelist | Submit | Opens the **Submit Files** dialog. See "Checking in files" on page 34. |
| | Revert | Reverts the selected file or files. See "Checking in files" on page 34. |
| | Move to Another Changelist | Opens a dialog that lets you select a different changelist. See "Moving a file to another changelist" on page 41. |
| | Shelve | Opens the Shelve dialog. See "Shelving files" on page 51. |
| | Diff Against Have Revision, or Diff Against another revision | Opens the workspace revision and have revision side by side in P4Merge, or opens the **Diff** dialog. See "Comparing files using diff" on page 48. |
| | Change filetype | Opens the Change Filetype dialog. See "Changing Helix server file types" on page 49. |
| | Lock | Locks a file to prevent others from checking it out while you are working on it. See "Checking out and editing files" on page 33. |
| | New Pending Changelist | Opens the **Pending Changelist** dialog. See "Editing changelists" on page 36. |
| | Refresh Pending Changelist List | Updates the **Pending** tool window or the pending changelist that you have selected. |
| Job in a pending changelist | Remove from Changelist | Removes the job from the changelist, but does not delete it. |

**To edit a changelist from the Pending Changelist dialog:**

1. Go to **View > Pending Changelists** to open the **Pending** tool window. Alternatively, in the status bar, click **1 Pending Change** or ***X* Pending Changes**.
2. Right-click the changelist in the viewer and select **Edit Pending Changelist *changelist name*** to open the **Pending Changelist** dialog.

In the **Pending Changelist** dialog, you can view workspace and user information at the top.

3. To limit who can view the changelist, select the **Restrict access to changelist** check box. See also "Restricting access to changelists" on page 41.
4. Provide a description for the changelist.
5. Under **Files**, right-click and select a menu option to do any of the following:
   - Select files for inclusion
   - Move files to another changelist
   - Revert files
   - Unshelve, delete, or view shelved files
   - Attach or view associated jobs
   - Perform a diff on a file pending submission by right-clicking the file and selecting **Diff Against Have Revision**. For more information, see "Comparing files using diff" on page 48.
6. Under **Jobs**, view and add associated jobs, or remove jobs.
7. Click **OK** to save your changes.

**To work with changelists from the Submitted tool window:**

1. Go to **View > Submitted Changelists** to open the **Submitted** window.
2. Right-click the changelist, a file, or an associated job and select any of the following actions.

| Item to right-click | Action | More information |
|---|---|---|
| Submitted changelist | Get Revision | Opens the **Get Revision** dialog. See "Retrieving files from the depot" on page 32. |
| | Edit Submitted Changelist | Opens the **Submitted Changelist** dialog. See "Editing changelists" on page 36. |
| | Refresh Submitted Changelist List, or Refresh Submitted Changelist | Updates the **Submitted** tool window or the submitted changelist that you have selected. |
| File in a submitted changelist | Diff Against Previous Revision | Opens both file revisions in P4Merge. See "Comparing files using diff" on page 48. |
| Job in a pending changelist | Remove from Changelist | Removes the job from the changelist, but does not delete it. |

**To edit a changelist from the Submitted Changelist dialog:**

1. Go to **View > Submitted Changelists** to open the **Submitted** tool window.

2. Right-click the changelist in the viewer and select **Edit Submitted Changelist *changelist name*** to open the **Submitted Changelist** dialog.

   In the **Submitted Changelist** dialog, you can view workspace and user information at the top.
3. To limit who can view the changelist, select the **Restrict access to changelist** check box. See also "Restricting access to changelists" below.
4. If needed, edit the description for the changelist.
5. Under **Files**, right-click a file and do any of the following:
   - Select **Diff Against Have Revision** to compare the submitted file against the have revision. For more information, see "Comparing files using diff" on page 48.
   - Select **File History** to open the **File History** dialog. For more information, see "Displaying the revision history of a file or folder" on page 51.
   - Select **Time-lapse View** to open the **Time-lapse View** window. For more information, see "Viewing file history with Time-lapse View" on page 63.
6. Under **Jobs**, view and add associated jobs, or remove jobs.
7. Click **OK** to save your changes.

## Restricting access to changelists

By default, all users can view a pending or submitted changelist, regardless of whether they are permitted access to the files in the changelist by the protections table. To prevent users from seeing a changelist, check the **Restrict Access to Changelist** option when you edit a pending or submitted changelist.

This option enables the following restrictions:

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

## Moving a file to another changelist

To move a file from its current changelist to another one, do one of the following:

- Right-click the file in Solution Explorer and select **Manage Files > Move to another Changelist…**.
- Right-click the file in the **Pending** tool window and select **Move to another Changelist…**.

In the dialog that opens, select the changelist you want to move the file to.

## *Setting changelist display preferences*

To minimize the time it takes P4VS to handle very large changelists, limit the number of files displayed in a changelist by setting the **Maximum number of files displayed per changelist** field in the P4VS preferences under **Tools > Options > Source Control.** See "Setting P4VS preferences" on page 14.

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 Helix server file badges).

# Resolving conflicting changes

Conflicts occur when you attempt to integrate a file into an existing codeline or to 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.

When you attempt to submit a changelist containing a file that must be resolved, a Helix server Command Error is returned: `Merges still pending—use 'resolve' to merge files.` When you return to the Solution Explorer, you will see a red question-mark badge next to the file icon in Solution Explorer  : (You may need to right-click the file icon and select **Refresh** to see the question-mark badge).

If there is a yellow triangle badge on any file, get the latest revision of that file by right-clicking it and selecting **Revisions > 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.

> **Note**
> In the P4VS **Resolve** dialog, **Target** is the file in your workspace and **Source** is the file in the depot.



## *Resolving multiple files*

When there are multiple files in a changelist that need to be resolved, it is recommended that you first try to resolve them automatically.

To resolve multiple files at once, automatically:

1. Select the files in Solution Explorer, then right-click and select **Copy/Merge > Resolve…**.
2. In the **Resolve** dialog, select **Auto resolve multiple files**.

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

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

   If you select this option, P4VS treats binary files like text files and attempts a textual merge between the source and target files.

4. Select a **Resolve method**:

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

5. (Optional) Select **Set as Auto Default** to set your selections as the default for auto-resolving multiple files.
6. Click **Auto Resolve**.
7. 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 "Resolving individual files" below.

## *Resolving individual files*

To resolve an individual file:

1. Select the file in Solution Explorer, then right-click and select **Copy/Merge > Resolve…**.
2. Select **Interactively resolve files one at a time**.

   The **Resolve** dialog displays the **Files to Resolve**. If you are resolving multiple files one at a time, select the file you want to resolve. The files are removed from this list as they are resolved.

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

If you select this option, P4VS treats binary files like text files and attempts a textual merge between the source and target files.

4. View the **Recommended action**.

   P4VS recommends an action, based on the differences and conflicts in the files selected. It also displays:

   - The common base file
   - The number of differences between the source and base file
   - The number of differences between the target and base file
   - The number of conflicts that would be present in the merged result.

5. Select a **Resolve method**:

   - **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 target file (in your workspace) and source file (in the depot).
   - **Run merge tool**: Opens your chosen merge tool, enabling you to edit the file and save the merged result.

6. Select any **Additional Actions** that apply:

   - **Open File**: Enables you to open either version of the file individually or the merged result file in any editor.
   - **Diff**: Opens your diff tool to diff the files with each other or with the base file. It also enables you to diff the source, target, and base file with the merged file.
   - **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 history of either file using the **Revision Graph** tool.

7. When the resolve is complete, check in the changes by submitting the changelist that includes the resolved file.

> **Note**
> The default diff and merge tool for P4VS is P4Merge. You can set diff and merge preferences, including configuring the diff and merge tool of your choice, on the **Helix Core - Diff/Merge** node under **Tools > Options > Source Control**.

# Reconciling offline work

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

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

1. 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.

   P4VS 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" on page 49 and "Defining a workspace view" on page 27.
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.

# Deleting files

To delete a file from the depot, you must delete it using Visual Studio, mark it for delete using P4VS, 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 **Delete**.

   P4VS asks if you want to mark the file for delete.
2. On the P4VS dialog, select the default pending changelist or a new changelist
3. Click **Yes**.

   P4VS marks the file for delete and it is placed in a changelist.
4. Submit the changelist containing the file. The file is deleted from the depot and your client workspace.

If you want to keep a file in your project but avoid adding it to Helix server control, use Ignore Lists. For more information, see "Excluding Files from Helix server Control" below.

# Excluding Files from Helix server Control

Your workspace may include files that you do not want to add to the Helix server repository, such as files used or generated by automated build processes.

You can use Visual Studio to exclude a file from a solution by right-clicking the file in Solution Explorer and selecting **Exclude from Project** in the context menu. If the file is under Helix server control, P4VS prompts you to mark the file for delete, and after submitting the changelist that includes the deletions, the file is removed from both the project and the Helix server repository.

You can also use *Ignore Lists* in P4VS to specify files or filetypes that you want to keep in your project but do not want to add to the Helix server repository. An **Ignore List** is a file in your local workspace directory that contains a list of file names or file types to ignore. For example, you can create an Ignore List called `.p4ignore` in your project folder that contains the following:

```
*.swp
*~
tmp/*
.p4ignore.txt
```

(Note that the **Ignore List** file itself is included in the list.)

You can add an **Ignore List** file at any level of the solution hierarchy in your workspace. If you set your `P4IGNORE` environment variable to the file name of the **Ignore List** file, P4VS will not mark the listed files and filetypes for add, nor will it prompt you to do so.

**Ignore List**s only affect commands that search for and *add* new files. If you have already marked a file for add, P4VS will no longer ignore it, even if it or its filetype appear in an Ignore List.

You can add Ignore Lists at any folder level in your workspace (or solution). P4VS applies the rules in the **Ignore List** at the deepest folder level relative to the file being checked, along with the rules in any **Ignore List**s found in parent folders (although you can use the `!` character to override higher-level rules.)

The syntax for ignore rules is not the same as Helix server syntax. Instead, it is similar to that used by other versioning systems:

- Files are specified in local syntax
- `#` at the beginning of a line denotes a comment
- `!` at the beginning of a line excludes the file specification
- `*` wildcard matches substrings

For example:

| | |
|---|---|
| `foo.txt` | Ignore files called `foo.txt` |
| `*.exe` | Ignore all executables |
| `!bar.exe` | Exclude `bar.exe` from being ignored |

While you can set your local `P4IGNORE` environment variable and add **Ignore List**s manually, P4VS provides preferences and context menu options to simplify the process of adding and editing **Ignore List**s.

## Setting Ignore List preferences

Go to **Tools > Options > Source Control > Helix Core - Ignoring Files** to set Ignore List preferences, including the **Ignore List** file name. The file name you enter in your preferences is set by P4VS as the local `P4IGNORE` environment variable and used for all of your **Ignore List**s. For more information about setting **Ignore List** preferences, see "Setting P4VS preferences" on page 14.

## Adding a file to an Ignore List

To add a file to an **Ignore List** in Solution Explorer, right-click the file and select **Manage Files > Add to Ignore List**. P4VS adds the file to the **Ignore List** in the current folder. If there is no **Ignore List** file in the current folder, P4VS creates one. P4VS denotes an ignored file with a gray circle glyph next to the file icon.

## Removing a file from an Ignore List

To remove a file from an **Ignore List** in Solution Explorer, right-click the file and select **Manage Files > Remove from Ignore List**. P4VS adds an exclusionary (`!`) line for the file in the **Ignore List** in the current folder, which overrides any **Ignore List**s in parent folders.

## *Editing Ignore Lists*

To edit an **Ignore List** in Solution Explorer, right-click any file in the same folder and select **Manage Files > Edit Ignore List**. P4VS opens the **Ignore List** file for edit. If there is no **Ignore List** file in the current folder, P4VS creates one. Use **Edit Ignore List** when you want to add file types using wildcard expressions.

# Comparing files using diff

You can compare file revisions using the diff tool associated with P4VS. The default diff tool is P4Merge, which is included with P4V. To associate a different diff tool, go to **Tools > Options > Source Control > Helix Core - Diff/Merge**. For more information, see "Setting P4VS preferences" on page 14.

**To diff two files or file revisions:**

1. In the Solution Explorer, **Submitted** tool window, **Submit Changelist** dialog, **Submitted Changelist** dialog, **Pending** tool window, or **Pending Changelists** dialog, right-click the file whose revisions you want to diff.

   You can also diff two file revisions from the **File History** tool window by dragging one revision row onto another.

2. Select one of the following:
   - **Diff > Diff Against...**: compare any two files or revisions of a file.
   - **Diff > Diff Against Have Revision**: compare the file version in your workspace against the depot revision that you retrieved most recently. This selection opens P4Merge (or your preferred diff tool, if it is not P4Merge) without first opening the **Diff** dialog.
   - **Diff Against Previous Revision** (from **Submitted** tool window only): compare the revision you selected against the version in the previous changelist. This selection opens P4Merge (or your preferred diff tool, if it is not P4Merge) without first opening the **Diff** dialog.
3. In the **Diff** dialog, specify the revisions of the files you want to diff:
   - **Path**: the two 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.

4. Click **Diff**. P4VS launches P4Merge (or your preferred diff tool, if it is not P4Merge), displaying the differences between the files at the specified revision.

For more information about diffing files with P4Merge, see the P4Merge help.

## Changing Helix server file types

Helix server file types determine how a file is stored in the depot and synced (retrieved) 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 file and choose **Manage Files > Change Filetype…**.

   The **Change Filetype** dialog is displayed.

2. Set the desired type and attributes and click **OK** to dismiss the dialog.

   If the file was not checked out, P4VS checks it out and makes the change.

3. Submit the changelist containing the file.

For details about file types and attributes, see the *Helix Core P4 Command Reference*.

## Renaming and moving Files

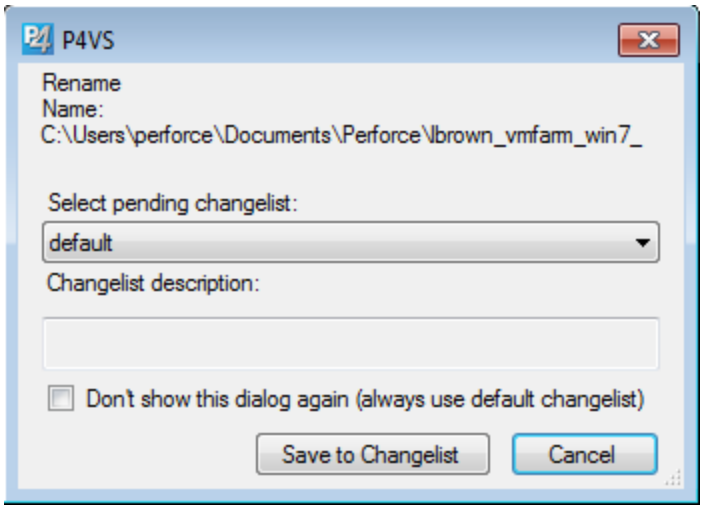
When you rename or move a file using the Visual Studio **Rename** option, P4VS prompts you to add the renamed or moved file to a changelist. When you do, P4VS automatically marks the new file name or location for add and the old file name or location for delete. When you submit the changelist, Helix server creates an integration record that links the renamed or moved object to its deleted predecessor, preserving its history.

To rename a file:

1. In Solution Explorer, right-click the file or folder you want to rename and select **Rename**.

   The file name in Solution Explorer becomes writable.

2. Type the new name.
3. When you leave the edit box, P4VS prompts you to add the renamed file to a changelist:




4. Select a pending changelist. The description defaults to the following, but you can change it:

```
\__path__\__old_filename__ to:
\__path__\__new_filename__
```

5. Click **Save to Changelist** to save the changes.
6. If there are code references to the renamed file in your project, Visual Studio asks if you want to rename all references.

   If you click **Yes**, Visual Studio renames all references, and P4VS prompts you to add the changes to a changelist. Follow the steps listed here to complete the process of submitting those changes to the Helix server depot.
7. Submit the changelist.

   The changelist includes **Add** operations for the new file name and **Delete** operations for the old file name.

   For more information about submitting changelists, see "Checking in files and working with changelists" on page 34.

To move a file from one location to another:

1. Right-click the file you want to move and drag it to the new location.
2. P4VS prompts you to add the moved file to a changelist.
3. Select a pending changelist. The description defaults to the following, but you can change it:

```
\__path__\__old_filename__ to:
\__path__\__new_filename_+
```

4. Click **Save to Changelist** to save the change.
5. Submit the changelist.

The changelist includes **Move/Add** operations for the new file location and **Move/Delete** operations for the old file location.

For more information about submitting changelists, see "Checking in files and working with changelists" on page 34.

> **Note**
> When you revert a rename or move operation in P4VS, Visual Studio continues to show the new file name or location despite the fact that Helix server has reverted it to the original name or location, unless you select **Update related projects when reverting moved files** in the P4VS preferences. For more information, see "Setting P4VS preferences" on page 14.

# Displaying the revision history of a file or folder

To display a file's revision history:

1. Open the **File History** tool window by doing one of the following:
   - Right-click the file or folder icon in Solution Explorer and choose **Revisions > Show History**.
   - Go to **View > File History**.
   - In the **Submitted Changelist** dialog, right-click a file and select **File History**.
2. View file revisions in the **File History** tool window by clicking the triangle to the left of the file name.
3. To view details, including changelist descriptions, select **Details**.
4. To view integration history, select **Integrations**.
5. To view label history, select **Labels**.
6. To diff two file revisions, drag one revision row and drop it onto the other.

   This launches P4Merge (or your preferred diff tool, if it is not P4Merge), which displays the differences between the two file revisions. For more information about diffing files with P4Merge, see the *P4Merge User Guide*.

# Shelving files

Shelving enables you to store copies of open files temporarily in the 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 indicated by a red triangle marked by a file icon:  . When the changelist is expanded, shelved files are listed under the **Shelved Files** node. They are indicated by a file icon with a badge, for example:  . The badge can be a check mark, an X, a plus sign (+), or an integration arrow, depending on the pending action before shelving.

When managing shelved files, note the following:

- **Basics**: To be shelved, a file must be checked out. However, you cannot unshelve a checked-out file.
- **Submitting shelved files**: As of Helix server 2013.1, you can submit a shelved file directly. For previous versions of Helix server, you must first unshelve a file to submit it, then delete the shelved copy. (Unshelving does not delete the shelved copy.)
- **Managing changelists**: You cannot move a shelved copy to another pending changelist. If you revert a file after shelving it, the copy remains shelved in the changelist until you delete it. Only the changelist owner can reshelve or delete files that are shelved in the changelist. For Helix server releases that predate version 2013.1, you cannot submit a changelist that contains shelved files; you must delete the shelved copies before submitting. Starting with Helix server 2013.1, you can submit shelved files directly, but your changelist must contain only shelved files.
- **File history**: No file history is created when you shelve or unshelve files.
- **Diffing**: You can diff shelved copies by right-clicking the shelved file in the **Pending** dialog (**View > Pending Changelists**) and selecting **Diff Against Source Revision** or **Diff Against Workspace File**.



# Shelving checked-out files

To shelve checked-out files in a pending changelist:

1. Open the **Shelve** dialog by doing one of the following:
   - Go to **View > Pending Changelists**. On the **Pending** dialog, right-click the changelist and select **Shelve…**.
   - In the Solution Explorer, right-click a file that is in a pending changelist and select **Shelve…**.
2. In the the **Shelve** dialog, select the files you want to shelve.
3. Select any of the following options that apply:
   - **Revert checked-out files after they are shelved**: The files in your workspace will revert to the head revision in the depot.
   - **Clear changelist of all previously shelved files before shelving**
4. Click **Shelve**.
5. When prompted, enter a description and click **OK**.

   P4VS shelves the file in the selected changelist or, if you are shelving files in the default changelist, creates a new changelist.

# Unshelving 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 in the changelist and select **Unshelve…**. P4V displays the **Unshelve** dialog.
2. Check the files you want to unshelve and click **Unshelve** and any other desired options. The shelved file is copied to your workspace and opened in the specified changelist.

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 select **Delete**. You can also right-click the pending changelist and select **Delete Shelved Files…**.

# Submitting shelved files

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

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

To submit shelved files in a pending changelist, right-click the changelist and choose **Submit Shelved Files…**.

# Working with streams

This chapter explains how to use P4VS with Helix server streams.

Before reading this chapter, review the "Streams" chapter in the *Helix Core Server User Guide* and the "Basic of Version Control" chapter in *Solutions Overview: Helix Version Control System*, which explain fundamental stream concepts.

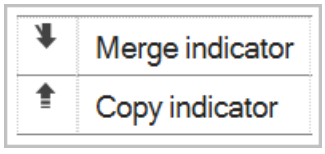
You may also find it helpful to see the P4V help (launch P4V from the context menu using **Views > View in P4V**, and click **Help** on the P4V toolbar).



## Using the Streams tool window

P4VS provides two ways to view streams graphically: you can use the **Streams tool window** directly in P4VS, or you can call the **Stream Graph**, a P4V component, from within P4VS. This topic discusses how to use the Streams tool window.

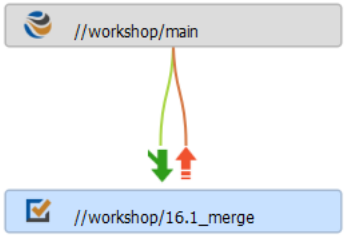
In the **Streams** tool window, status indicators between streams tell you which streams have changes to contribute and where the changes can be copied or merged:




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 next to a stream indicate that you must update it by merging down from its parent, after which you can copy up changes to the parent.




Right-clicking on a stream in the **Streams** tool window shows the available copy and merge actions that you can perform. If you need to work in another stream to complete an action, you are prompted to switch workspaces, create a new workspace, or select a workspace from an available list depending on the existing workspaces that are available for use with the target stream. From there you can preview the copy or merge operation and complete it. After the copy or merge is done, you are prompted to select a changelist (if the preference is set for changelist prompts) and then to save or submit that changelist. When the merge or copy workflow is complete, your connection changes back to the original workspace that was in use if the workspace was switched during the merge or copy process.

## Editing Ignore Lists

To edit an **Ignore List** in Solution Explorer, right-click any file in the same folder and select **Manage Files > Edit Ignore List**. P4VS opens the **Ignore List** file for edit. If there is no **Ignore List** file in the current folder, P4VS creates one. Use **Edit Ignore List** when you want to add file types using wildcard expressions.

## Displaying and searching for streams

To display the streams defined for the Helix server depot to which you are connected in P4VS:

1. Go to **View > Streams** to open the **Streams** tool window.
2. Search for streams using the filter fields.

   You can filter by any combination of the following:

   - Depot (requires an entry)
   - Owner
   - Name
   - Parent
   - Type

Use standard Helix server syntax (`//streamdepot/stream`). For more information, see the *Helix Core Server User Guide*.

Note that because this tool window provides a hierarchical view of streams, you may see parent streams that do not match the filter. These are included in the list to show the hierarchy of the streams all the way to the related mainline, but are grayed out.

To change the order in which columns are displayed, drag the column headings right or left.

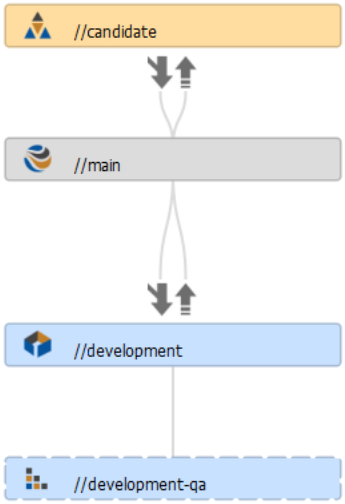
The details of a selected stream are displayed in the lower pane.

## Using the Stream Graph

P4VS provides two ways to view streams graphically: you can use the **Streams** tool window directly in P4VS, or you can call the **Stream Graph**, a P4V component, from within P4VS. This section discusses how to use the **Stream Graph**.

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. For example:




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

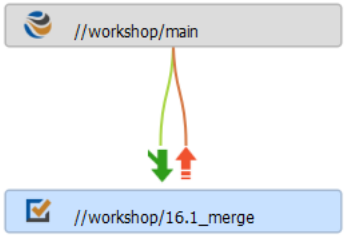
**Merge indicator:**

**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 next to a 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.

## Accessing the Stream Graph from P4VS

Go to **File > Helix > Views > Stream Graph** or right-click in the Solution Explorer and select **Views > Stream Graph**.

> **Note**
> The **Stream Graph** is a P4V component. When you are working in the **Stream Graph**, you are working in P4V.

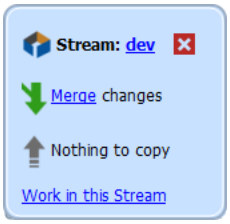
## Setting Stream Graph display options

Select display options in the **Graph View Options** dialog:

1. In the **Depot** drop-down list, select the depot containing the streams you want to view. By default, the graph shows the stream containing the files you are currently working in.
2. To select the streams you want displayed in the graph, click **Select Streams** and choose the display option, or check the individual streams that you want displayed in the graph. You may need to expand the tree within the dialog pane to view the streams you want to select.
3. Click **Apply Filter**. The stream graph displays the streams that you specified.
4. (Optional) In the **Graph Navigator** dialog, configure the size of the stream graph display and select which portion of the stream graph to view. Use your mouse or cursor keys on the navigator pane to select the portion of the image you want to view.

## Displaying stream status

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



# Working in a stream

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

- Double-click the stream and select **Work in this stream**.
- Drag the workspace icon ( ) from the stream you are working in to the one you want to work in.

> **Important**
> In order to switch streams in P4VS using the **Stream Graph**, you must set your P4V stream operations preference to **Use the same workspace and switch it between streams**.

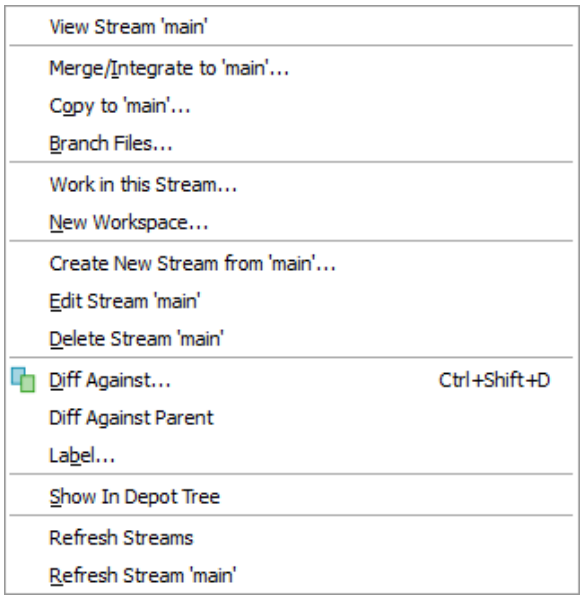
If you have not set this preference, a warning dialog pops up when you try to switch streams and asks you to switch workspaces or create a new one. If you then click the **Switch Workspaces** button, the dialog closes, as does the **Stream Graph**, with the workspace unswitched. If you click the **New Workspace** button, the **Workspace:New** dialog opens. You can create a new workspace, but the dialog and **Stream Graph** close without switching workspaces in P4VS.

If you do not want to use the same workspace when switching streams in P4VS, you must open a new connection to Helix server to select a new workspace.

For more information about setting P4V preferences, see "Configuring P4V Preferences" in the P4V help (launch P4V from the context menu using **Views > View in P4V**, and click **Help** on the P4V toolbar).

# Other actions you can perform with the Stream Graph

When you right-click a stream in the **Stream Graph**, you see the following options:



To learn about these streams options, see the P4V help (launch P4V from the context menu using **Views > View in P4V**, and click **Help** on the P4V toolbar).

# 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.

## *Merging down*

To merge changes down to a less stable stream:

1. Go to **File > Helix > Copy/Merge > Merge to Stream…** or right-click in the Solution Explorer and select **Copy/Merge > Merge to Stream…**.

   When you merge down or copy up, you must be working in the target stream.
2. In the **Merge** dialog, select the **Source Stream** (the stream you want to merge down changes from). This must be a parent of the target stream.
3. (Optional) Click **Preview** to view the merge results.
4. Click **Merge**.
5. If necessary, resolve the merges manually, then submit the resulting changelist.

If you want to merge changes between streams without working in the target stream, open the **Streams** tool window, right-click a stream that shows a pending **Merge** indicator, and select **Merge to '*streamname*' from parent**.

If you want to use advanced options when merging changes between streams, launch the Stream Graph, context click the stream you want to merge down to, and select **Merge/Integrate to '*streamname*'**. You can also use P4V or the Helix server command-line client. For more information about the full set of **Merge** options, see the P4V help (launch P4V from the context menu using **Views > View in P4V**, and click **Help** on the P4V toolbar).

## *Copying 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. Go to **File > Helix > Copy/Merge > Copy to Stream…** or right-click in the Solution Explorer and select **Copy/Merge > Copy to Stream…**.

   When you merge down or copy up, you must be working in the target stream.
2. In the **Copy** dialog, select the **Source Stream** you want to copy from.
3. (Optional) Click **Preview** to view the copy results.

4. Click **Copy**.
5. Submit the resulting changelist.

If you want to copy changes between streams without working in the target stream, open the **Streams** tool window, right-click a stream that shows a pending **Copy** indicator, and select **Copy to '*streamname*' from parent** or **Copy to parent from '*streamname*'**.

If you want to use advanced options when copying changes between streams, launch the **Stream Graph**, context click the stream you want to copy up to, and select **Merge/Integrate to '*streamname*'**. You can also use P4V or the Helix server command-line client. For more information about the full set of **Copy** options, see the P4V help (launch P4V from the context menu using **Views > View in P4V**, and click **Help** on the P4V toolbar).

## *Propagating change between unrelated streams*

To propagate change between streams that are not directly connected, use P4V or the Helix server command-line client.

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 that you cannot reparent a task stream.

For more information, see "Merging Down and Copying Up between Streams" in the P4V help (launch P4V from the context menu using **Views > View in P4V**, and click **Help** on the P4V toolbar).

# Using other Helix server features

This chapter discusses how to take advantage of other Helix server features available from within P4VS.



## Viewing integration history in the Revision Graph

The **Revision Graph** displays file integration history, showing when a file was added, branched, edited, integrated, and deleted.

### *Launching Revision Graph*

Right-click a file or folder in the Solution Explorer or go to **File > Helix** and select **Views > Revision Graph**.

> **Note**
> The **Revision Graph** is a P4V component. When you are working in the **Revision Graph**, you are working in P4V.

### *Reading the Revision Graph*

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


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

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

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

## Navigating the Revision Graph

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

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

To navigate the diagram:

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

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

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

To diff two revisions, drag one revision to another or select the revisions, then 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**.

## Filtering the Revision Graph

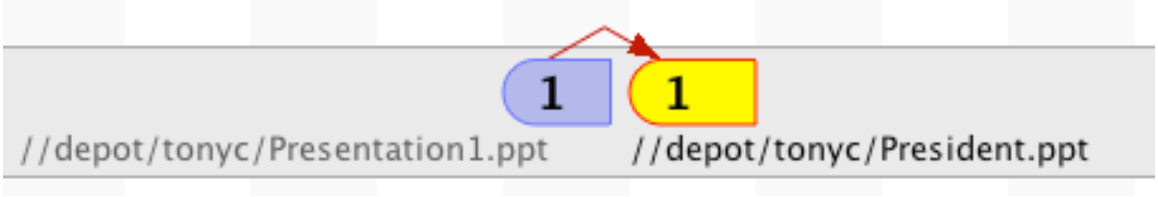
To reduce the detail displayed in the main window, you can filter the information. To remove a file or folder from the main window, uncheck it in the **File Filter** tree.

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

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

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

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




## *Displaying details*

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

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

## *Displaying Time-lapse View*

To open the Time-lapse View window, do any of the following:

Right-click in the Solution Explorer and select **Views > Time-lapse View**.

Go to **Files > Helix** and select **Views > Time-lapse view**.

In the **Submitted Changelist** dialog, right-click a file and select **Time-lapse View**.

> **Note**
> **Time-lapse View** is a P4V component. When you are working in **Time-lapse View**, you are working in P4V.

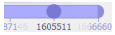
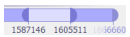
## *Controlling the display*

The following options are available on the toolbar:

| | |
|---|---|
| **Mode** | Determines how many revisions are displayed. Options are:<br>■ Single revision: one revision at a time is displayed<br>■ Incremental diffs: two adjacent revisions are displayed, with changes highlighted<br>■ 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. |
| **Branch History**  | Toggles inclusion of branching (integration) history. |
| **Find**  | Search text |
| **Go To**  | In single revision mode, go to specified line number. |
| **Go to Next diff**  | Go to next modification |
| **Go to Previous diff**  | Go to previous modification |

| | |
|---|---|
| **Line Ending**  | Specifies how line endings and whitespace are treated to determine differences |

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. |

# Finding code changes and references with CodeLens

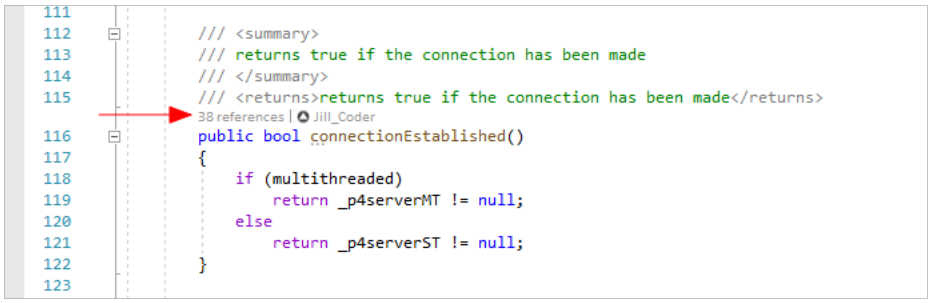
With Microsoft Visual Studio 2019 version 16.1, P4VS supports the CodeLens feature. With the relevant CodeLens option enabled (which is the default), you can view the names of users who made changes to classes, functions, and methods throughout the code and get a preview of up to five changelists. You can also open a changelist, the file history, and the Time-lapse View.

> **Note**
> The **Time-laps View** link is only available if P4V is installed.

If needed, you can modify the configuration under **Tools > Options > Text Editor > All Languages > CodeLens > Show Submitted Changelists (Helix)**.

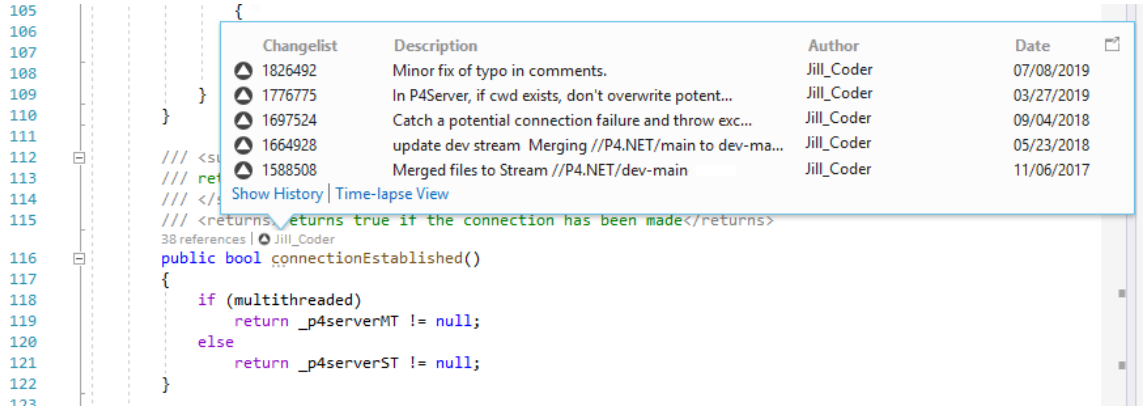
**To use the CodeLens feature with P4VS:**

1. Above a class or function, move the pointer over the gray user name, as indicated in the following figure.




A tooltip with additional information about the last changes appears.

2. To open a callout displaying the latest five changes, as shown in the following figure, click the user name.




3. To view more information on any of these changes, do one of the following:
   - Move the pointer over the change to view the full description.
   - Double-click the change to open the **Submitted Changelist** dialog. For details, see the "Checking in files and working with changelists" on page 34 section.
4. To view more information on the file itself, click the Show History or Time-lapse View links.

# Viewing a project in P4V, the Helix Visual Client

P4V is the dedicated visual client application for Helix server. It provides a rich interface for managing your projects under source control.

To view a project or file in P4V, right-click the project or files you want to view in P4V and select **Views > View in P4V**.

For more information about P4V, see *P4V User Guide*.

# Using jobs (defect tracking)

Jobs enable you to record requests for work. You can associate jobs with changelists to track the work done to fulfill the request. When you submit the changelist, the job can be closed.

## *Creating jobs*

1. Go to **View > Jobs**.
2. In the **Jobs** tool window, right-click anywhere in the job list pane and select **New Job…**
3. Fill in the **Job** form.

   The fields that appear on the **Job** form depend on the customizations set up by your Helix server administrator. For more information, see *Helix Core Server Administrator Guide*.
4. Click **OK**.

## *Editing jobs*

1. Go to **View > Jobs**.
2. In the **Jobs** tool window, right-click a job row and select **Edit Job…**.
3. Update the **Job** form.

   The fields that appear on the Job form depend on the customizations set up by your Helix server administrator. For more information, see *Helix Core Server Administrator Guide*.
4. Click **OK**.

## *Displaying jobs*

To view jobs:

1. Go to **View > Jobs**.

   In the **Jobs** tool window, enter search terms in the **Keywords** field or the depot directory path in the **Folder/file** field.

   For keyword syntax, see "Filtering Expressions" on the next page.

   Use the **Folder/file** field when you know the location of a file that is included in an associated changelist. Enter the directory path using Helix server syntax (`//depot/folder/folder/filename` or `//depot/folder/…`)
2. Click **Filter**.
3. Click a job row to view details about the job.

To change the order in which columns are displayed, drag the column headings right or left to the desired position. To sort by column, click the sort arrow on a column heading.

## Associating changelists with jobs

To add a job to a pending changelist:

1. Open the **Submit** dialog.
2. Select a changelist in the **Link jobs to changelist** list.

   If the job you want is not on the list, add it by clicking **Browse...**. In the **Jobs Browser**, find and select the job you want. For keyword syntax, see "Filtering Expressions" below.
3. Specify the **Job status upon submit**: `open`, `suspended`, or `closed`.

You can also add a changelist to a job by editing the job. For more information, see "Editing jobs" on the previous page.

## Filtering Expressions

Valid job filtering expressions are as follows:

| Syntax | Description | Example |
|---|---|---|
| `word word word` | Words separated by spaces indicate that the job must contain all the words in the string in any of the job fields to be included in the filter. Spaces represent the boolean "and". | `filter file mailbox`<br>Displays jobs containing all the words "filter", "file", and "mailbox" in any of the job fields. |
| `word \| word \| word` | Displays jobs that contain any of the specified words. Pipes represent the boolean "or". | `filter\|file\|mailbox`<br>Displays jobs containing the words "filter", "file" or "mailbox". |
| `^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`<br>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`<br>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`<br>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*`<br>Displays jobs in which the value of the field `owner` contains the substring `ed`, including such values as `Ted`, `Edk`, and `Fred`. |

# Using labels

Labels can be used to mark important file revisions, such as the set of file revisions used to build a particular software release. You can use labels to specify groups of related file revisions when you get file revisions (sync), compare file revisions (diff), and integrate (merge, copy, and branch).

To use labels, you first define the label and then apply the label to file revisions in the depot.

## *Creating and editing labels*

You must use P4V, the Helix Visual Client , or `p4`, the Helix server command-line client, to create and edit labels. For more information, see the P4V help or the *Helix Core Server User Guide*.

## *Labeling files*

You must use P4V, the Helix Visual Client , or `p4`, the Helix server command-line client, to apply labels to files. For more information, see the P4V help or the *Helix Core Server User Guide*.

## *Displaying and searching for labels*

To display the labels defined for the Helix server depot to which you are connected in P4VS:

1. Go to **View > Labels** to open the **Labels** tool window.
2. To search for labels, use the filter fields.

   You can filter by any combination of the following:

   - Owner
   - Label name
   - File path

     Use standard Helix server syntax (`//depot/folder/folder/filename` or `//depot/folder/...`). For more information, see the *Helix Core Server User Guide*.

To change the order in which columns are displayed, drag the column headings right or left to the desired position. To sort by column, click the sort arrow on a column heading.

3. To view details about a label, such as the owner, description, and view, select the label row and click **Details** in the lower pane.
4. To view a list of files in a label, select the label row and click **Files** in the lower pane.

## *Retrieving files by label*

To retrieve a file revision in a label:

1. Right-click the file in Solution Explorer and select **Revisions > Get Revision**.
2. Select **Specify revision using: Label** and browse for the label.
3. (Optional) Select **Remove files from workspace if they are not in label** to ensure that your workspace contains only the labeled file revisions.

# Working with reviews in Swarm

Helix Swarm is a powerful and flexible code review and collaboration solution that helps teams ship quality software faster. Swarm enables review of code and other assets before or after commit and can be customised to fit into various workflows. Swarm stores all of its metadata including reviews, projects and comments in Helix server, which makes it an attractive solution since it doesn't require backing up an external database. For more about using and installing Swarm, please see the *Helix Swarm Guide*.

## *Workflow of a review*

Below is the happy path workflow for a Swarm review. There are more permutations and variations that are described in the Swarm documentation.

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 Helix server.
2. **Request a review**: For pre-commit code reviews, the Swarm solution uses Helix server shelving technology to get the content to Helix server. For post-commit code reviews, content committed to Helix server is added to a review. In both cases, a Swarm review is created with an id, a description, a set of files and other meta-data including the author, reviewers and comments made on the review.
3. **Provide review feedback**: Reviewers can comment on files or on individual lines of files using Swarm. Reviewers can also add follow-up tasks that the author would be asked to address before the review could be closed.

4. **Request revisions**: If the reviewers find the review needs more work, which is often the case, they can change the state of a review to **Needs Revision**, thereby notifying the author that the review is back in their court.
5. **Request further review**: Authors can request further review of their review content changes and update any of the tasks they were asked to complete, thereby notifying the reviewers that they are ready for more of their feedback.
6. **Approve or reject review**: Reviews can be approved or rejected using Swarm. Once a review is approved or rejected, it is considered closed.
7. **Commit the review.** For pre-commit reviews, authors can commit reviews using their Helix server clients such as P4V or P4VS. For this scenario, committing a pre-commit code review is synonymous with submitting the changelist associated with the review. They can also optionally use Swarm to commit pre-commit reviews.

## Setting up the Swarm integration

A minimum requirement for the P4VS integration is to run Swarm version 2014.4.

None of the new features for Swarm are available unless the Swarm integration is turned on. This integration needs to be turned on for each Helix server server. In order to make P4VS enable the Swarm features, the Helix Core administrator must run the **`p4 property`** command for the Swarm URL. This will tell the Helix server server the Swarm URL. The P4VS integration uses this URL when making API requests to the Swarm server.

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 the Swarm server.

If you are testing the Swarm integration, you may wish to set the property for a specific user. For example, to enable the Swarm integration for the user *username*:

```
p4 property -a -u _username_ -n P4.Swarm.URL -v https://_
swarm.yourcompanydomain.com_
```

Similarly, you can enable the Swarm integration for a specific group of users. For example, to enable the Swarm integration for the group *group*:

```
p4 property -a -g _group_ -n P4.Swarm.URL -v https://_
swarm.yourcompanydomain.com_
```

# Authentication with Swarm

P4VS uses the user's existing Helix server ticket to communicate with Swarm. If you get authorization errors, ensure that the **Use IP-specific tickets when logging in** is disabled. This is synonymous with using the `-a` option with the `p4 login` command so that the ticket can be used on any machine.

# Swarm integration features

Once the Swarm integration is enabled a number of new features are available in P4VS including new context menus, review request and update dialogs, badging on pending and committed changes, as well as **Review ID** and **Review State** columns.

# Request a review

Reviews can be requested 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.

Pre-commit code reviews are a more popular approach since 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 warrant that the author follow on with more submitted changes when wanting to make the updates recommended by the reviewers. Development branches are well-suited for the post-commit review process.

## Request a review from a pending changelist

To request a review from a pending changelist, go to **View > Pending changelists**, select the changelist and choose the **Request New Swarm Review…** from the context menu. Note that if the changelist is already part of a Swarm review, this option is not available.

The **Request New Swarm Review** dialog displays a list of files to be shelved in order to request the review. If the changelist already has shelved files, the dialog also lists these already shelved files. The aggregate of the shelved files comprises the review. The review must have a description, which defaults to the changelists' description. The dialog offers additional options including: reviewers, reverting checked out files after they are shelved, not shelving unchanged files, and opening the review in Swarm.

Once the review has been requested, the pending changelist is badged with a Swarm icon and P4VS updates the **Review ID** and the **Review State** fields with their values from Swarm.

It is a best practice for the author to keep this pending changelist for subsequent updates to the review. This same changelist can be used by the author to submit the review. If the review is rejected or the review is committed from Swarm, then the author 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, go to the **Submitted changelist** tab, select the changelist and choose the **Request New Swarm Review…** option from the context menu. Note that if the changelist is already part of a Swarm review, this option is not available.

The **Request New Swarm Review** dialog displays the files that to be added to the review. The review must have a description, which defaults to the changelists' description. The dialog offers additional options including: reviewers and opening the review in Swarm.

Once the review has been requested, the pending changelist is badged with a Swarm icon and P4VS updates the **Review ID** and the **Review State** fields with their values from Swarm.

## *Update Swarm 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, P4VS provides an option to update an existing Swarm review.

### Update a Swarm review from a pending changelist

To update a review from a pending changelist that is associated with the review, go to **View > Pending changelists**, select the changelist and choose the **Update Swarm Review '*xxxx*"…** option from the context menu, 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. You can also update the review description at this time. The dialog offers additional options including: reverting checked out files after they are shelved, not shelving unchanged files, and opening the review in Swarm.

### Update a Swarm review from a submitted changelist

To associate a submitted changelist with an existing Swarm review, select the submitted changelist and choose **Add to Swarm Review** context menu option.

The **Add to a Swarm Review** dialog displays a list of files to be added to a review. The dialog has a field where you can enter the review id of the review to which you'd like to add these files. Type in the review id in the **Update Review** field and click the **View Review Description** button if you want to see a preview of the review's description in order to verify that this is in fact the review you'd like to add these files to. The dialog offers an additional options to open the review in Swarm.

## *Open review in Swarm*

If you leave the **Open Review in Swarm** checkbox option selected in the **Review Request** or **Review Update** dialogs, then P4VS launches Swarm to the review page in a new tab in Microsoft Visual Studio (you can configure this to open in an external browser instead). This serves as confirmation that the review has been created or updated.

If a pending or submitted changelist is already associated with a review, context click the changelist and select **Open Review '*xxxx*' in Swarm…** to open the associated review, where *xxxx* is the id of the associated review.

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

## *Review Id and Review State columns*

P4VS will add a **Review Id** and **Review State** column to both the submitted and pending changelist tabs for connections that have the Swarm integration enabled.

If you are connected to a Helix server server with the Swarm integration enabled and do not see the columns, right click on the header row and select these fields.

# Glossary

## A

### access level

A permission assigned to a user to control which commands the user can execute. See 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.

### 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 that you are changing in your client workspace that is checked out. If the file is not checked out, opening it in the file system does not mean anything to the versioning engineer.

### 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 additional intelligence 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

For complete licensing information pertaining to P4VS, the Helix Plugin for Visual Studio, see the license file at https://www.perforce.com/perforce/doc.current/user/p4vslicenses.txt.