HelixCore

P4Admin User Guide

2020.1

April 2020

PERFORCE

www.perforce.com

Contents

How to use this guide

This guide tells you how to use P4Admin, a GUI for administrating Helix server connections. It is intended for anyone using P4Admin to perform basic Helix server administrative tasks. Access to the complete set of administrative tools requires the Helix Core server command-line client, also referred to as `p4`.

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. ▪ `p4 [g-opts] streamlog [ -l -L -t -m max ]` ***`stream1`*** `...` means `1` or more stream arguments separated by a space ▪ See also the use on `...` in Command alias syntax in the *Helix Core P4 Command Reference* **Tip** `...` has a different meaning for directories. See Wildcards in the *Helix Core P4 Command Reference*.
element1 | *element2*	Either *element1* or *element2* is required.

Feedback

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

Other documentation

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

Administering Helix server using P4Admin

P4Admin provides a graphical user interface for performing basic Helix server administration tasks. This chapter discusses the following topics:

To access P4Admin from within P4V, go to **Tools > Admininstration**.

Home page

The Administration home page enables users with `super` and `admin` permissions to view and use the following:

- **Alerts:** displays important messages, such as that your support has expired.
- **Server information:** displays details about the server to which you are connected.
- **Disk space usage:** displays details about server disk space usage.
- **Security level:** displays authentication level required by users who access the server.
- **Account management shortcuts:** displays links to common tasks.
- **User licenses:** displays details about license count and expiration.

- **Inactive users:** displays details about users who have not accessed their account in the period of time you specify.
- **Triggers:** displays details about the triggers in use.

Server connections

To administer a server, you must first connect to it as a user that has been granted the superuser privilege for the server. (If your user does not have the superuser privilege, you can still administer users and groups, but you cannot manage permissions or define depots.)

To connect to a server:

1. Choose **Connections > Open Connection**. The **Open Connection** dialog opens.
2. In the **Open Connection** dialog, enter the Helix server name and port number for the connection using the following syntax: `server_host:port_number`
3. In the **User** field, enter your user name.
   - To browse for a particular user, click the **Browse** button and select the user from that list.
   - To create a user, click **New** and fill in the appropriate information.
4. Click **OK**.

   P4Admin connects to the specified Helix server and displays a new instance of its main window.

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

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

To connect to Helix server using a connection that you have used previously:

Do one of the following:

- If P4Admin is already running, go to **Connections > Open Recent** and select the connection.
- When you launch P4Admin, select the connection from the **Connections** drop-down in the **Open Connection** dialog. The **Connections** drop-down lists recent and favorite connections.

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

To administer a server:

- Activate the connection by clicking its entry in the **Connections** pane.
- To remove a connection, right-click it and select **Close Connection**.

P4Admin retains all connections that you define, so you do not need to reenter them the next time you launch the tool.

Managing depots

If you have `super` user permission for the Helix server instance to which you are connected, you can manage the depots that it contains. Specifically, you can:

- "Display or edit depot details" on the next page
- "Create a depot" below
- "Delete a depot" on page 10
- "Obliterate files from a depot" on page 10

For detailed steps, see the individual sections.

Create a depot

1. Select **File > New > Depot**.
2. In the **New Depot** dialog, enter a name for the depot and click **OK**.
3. In the **Depot** window, provide the following information:
    - **Owner:** The user who owns the depot. By default, this is the user who created the depot.
    - **Description:** A short description of the depot's purpose. Optional.
    - **Depot type:** Can be any of the following:
        - **local:** (Default) Writable. Files reside in the server's root directory and are managed directly by the server. By default, there is one local depot named depot on every Helix server installation.
        - **stream:** Writable. Contains streams, a type of branch that includes hierarchy and policy.

If you select this depot type, you can also modify the stream depth, which is the optional depth to be used for stream paths in the depot. The depth specifies the number of slashes following the depot name of a stream. By default, the stream depth is 1, matching the traditional stream name. You cannot update this value once streams or archive data exist in a depot. When you use P4Admin to create a depot of type stream, the maximum depth of the stream is 10.

- **spec:** Automatically archives edited forms. For more information, see Working with spec depots in the *Helix Core P4 Command Reference*.

  If you select this depot type, you can also:

  - Add specification mapping, which is an optional description of which specs should be saved, expressed as a view.
  - Specify a file extension, which is appended to versioned specifications.
- **remote:** References files that reside on other server and cannot be written to. For more information, see Working with remote depots in the *Helix Core P4 Command Reference*.

  If you select this depot type, you can also specify the mapped portion of the remote server. By default, this is `//...`. The map points to a location in the remote depot's physical namespace, for example `//depot/new/rel2/...`. This directory will be the root of the local representation of the remote depot.
- **archive:** Used in conjunction with archiving (`p4 archive`) and restoring (`p4 restore`) to facilitate offline (or near-line) storate of infrequently accessed revisions, typically large binaries.
- **unload:** Holds infrequently used metadata (about old client workspaces and labels) that has been unloaded (with the `p4 unload` command). For more information, see the *Helix Core Server Administrator Guide*.
- **graph**: For storing repos in one or more graph depots.

  - **Storage of location for versioned files:** By default, the storage location is relative to the server root, but you can use an absolute path to stare files in a location that is not located under the server root.

4. Click **Apply** to save your changes and keep the window open or **Ok** to save your changes and close the window.

Display or edit depot details

1. Select **View > Depots**. The **Depots** tab opens in the right pane.

   The left side of the tab shows the depot tree. The right side shows the files that are located in the folder you have selected in the depot tree. The two sub-tabs at the bottom provide additional information:

- If and by who a selected file is checked out
- A preview of the file

2. To view additional information on a depot, in the **Depot** tab, right-click the depot and select **View Depot '<depot-name>'**.

   The **Depot** window opens, displaying information such as the depot name, date last modified, owner, description, type, and storage location for versioned files.

3. Optionally, click:
   - **Edit** to modify the **Owner**, **Description**, or **Storage location for versioned files** fields.

     All other fields are read-only for existing depots.

     When done with your edits, click **Apply** to save your changes and keep the window open or **OK** to save your changes and close the window.
   - **Print** to send the information to a printer of your choice. In the **Print** dialog, select the printer and click **Print**.
4. Click **Close**.

Delete a depot

1. In the **Depot** tab, right-click the depot you want to delete and select **Delete Depot '<depot-name>'**.
2. When prompted for confirmation, click **Yes**.

Obliterate files from a depot

> **Warning**
> Obliterate with extreme caution. Obliteration permanently removes *all* traces of the specified files from the Helix server, including revision records and metadata (such as references in labels and client workspace specifications). Files in client workspaces are left untouched, but they are no longer recognized as being under Helix server control.

1. In the **Depot** tab, right-click a file and select **Obliterate**.
2. In the **Obliterate** dialog that opens, click **Add** to browse to the file or files you want to delete.
3. Select whether you want to obliterate all revisions, all revisions up to a specific revision, or all revisions between two specific revisions.
4. Optionally, click **Preview** to see the effects of obliterating the selected files. For example:

```
//flow/D1/a/that.txt#1
//flow/D1/a/this.txt#1
```

```
Would delete the following:
* 2 record(s) from the revision database (0 archive record(s) purged)
* 4 record(s) from the have database
* 0 record(s) from the opened files database
* 12 record(s) from the integration database
* 0 record(s) from the label database
```

5. Click **Obliterate** to confirm.

Managing users and groups

All users can display lists of users and groups using P4Admin, but only users with *admin* or *super* permissions can make changes. For details about access levels, refer to the *Helix Core Server Administrator Guide*.

To view the users defined for the server to which you are connected, select **Tools > Administration** and click the **Users & Groups** tab.

To display details about a user, click the desired user specification. The details about that user are displayed at the bottom of the **Users** tab.

To create a user:

1. In P4V, select **Tools > Administration**. P4Admin opens.
2. Choose **File > New > User**.

   The **User** dialog opens.
3. Enter user information as follows. Click **OK** after making your entries.

User	The Helix server user name.
Password	The password (if any) required for the user to connect to the server.
Email	The user's email address.
Full name	The user's real name.
Job view	(Optional) Criteria specifying which jobs are automatically included on any new changelists created by the user Example: `User=bruno Status=open` For details, refer to the description of the `p4 jobs -e` flag in the *Perforce Command Reference*.

Reviews	Files of interest to the user, specified using depot syntax When changelists that affect the file are submitted, the user receives email notification of the change. Note that you can also specify files of interest by clicking the **Reviews** tab.
Groups	Groups to which the user belongs To add the user to a group, enter the name of the group in the **Group** field and click **Add**, or click **Browse**, select the group, and click **OK**.

To change your password: Select **Administration > Change Password**.

To display the groups to which a user belongs, expand the user in the **Users** pane. **To display the users in a group**, expand the group in the **Groups** pane.

To edit a user or group, right-click the user or group you want to edit and select **Edit**.

To see the areas of the depot tree to which a user has access, right-click the user, and select **Show Permissions**. The Permissions tab is displayed, with the selected user highlighted.

To add a user to a group, drag the user from the **Users** pane to the desired group. **To remove a user from a group**, right-click the user in the group and choose **Remove**.

To edit the groups to which a user belongs:

1. Right-click the user and select **Edit User**.

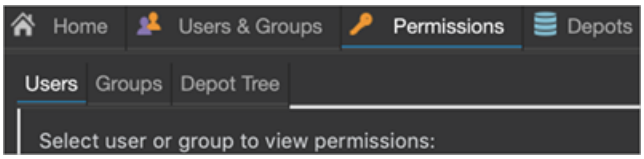
   The **User** dialog is displayed.
2. Edit the list displayed in the **Group membership** field.

Managing permissions

View permissions

The Permissions tab displays tabs under it for **Users**, **Groups**, and **Depot Tree**.

To display the files and folders to which a user has access, click the desired user on the **Users** tab.



To display the files and folders to which users in a group have access, click the desired group on the **Groups** tab.

To display the groups and users that have access to a file or folder, click the file or folder on the **Depot Tree** tab.

To see which lines of the protections table control access to a user, group, or area of the depot, click the user, group or folder of interest. The corresponding line in the protections table is highlighted. (If a user or group is neither granted nor denied access to a path by means of any entries in the protections table, the depot path displays "no access" and the "granted to" field is blank.)

To filter out lines in the right-hand pane, use the **Access Level** sliders to set the lowest and highest levels. The areas of the depot associated with the highlighted range of access values are displayed.

To see only those permissions that apply to a user's workstation, enter the IP address of the workstation in the **Host IP filter** field. For example, permissions lines with a host value of `92.168.*.*` and `192.168.1.*` both apply to a workstation at `192.168.1.10`.

To show files in the Depot Tree, click **Show files**.

> **Note**
> Virtual streams do not appear in the Depot Tree on the **Permissions** tab. Virtual streams map their parent's paths, and permissions for virtual streams are therefore always set for the parent's paths.

Edit the protections table

The protections table is displayed in the bottom pane of the screen. It is a representation of the table used by the **`p4 protect`** command, with exclusionary lines shown in red. For more information on the **`p4 protect`** command, see `p4 protect` in the *Helix Core P4 Command Reference*.

To edit the protections table, use the built-in editor or click  to edit the protections table as text.

To deny access to a specific portion of the depot to a user or group, use an exclusionary mapping. Place a dash (`-`) in front of the path in the **Folder/File** field. Exclusionary mappings apply to all access levels, even though only one access level can be selected in the **Access Level** field.

The following table describes the fields in the protections table.

Access Level

The permission being granted. Each permission level includes all lower-level permissions, except for **review**.

- **super**: Grants access all commands and command options
- **admin:** Permits those administrative commands and command options that don't affect server security
- **write**: Lets users submit open files
- **open**: Lets users open files for add, edit, delete, and integrate
- **read**: Lets users sync, diff, and print files
- **list**: Lets users see names but not contents of files. Users can see all non-file related metadata, such as workspaces, users, changelists, and jobs.
- **review**: Allows access to the `p4 review` command. This leve is intended for automated processes. It implies `read` access.
- **##:** Adds a comment line to the protections table. For example:

```
## robinson crusoe
write user *   10.1.1.1   //depot/test/...
```

Note

For stream specifications, you can use P4Admin to set stream spec access levels. For example, you can set an access level for a group or user, and then limit that access level to a subset of depots. Similarly, you can set an access level for a group and then remove it from a user in that group. The access levels for a stream spec are:

Access Level	Meaning the group or user can ...	Implies the lesser permission(s)
`writestreamspec`	submit or modify	`openstreamspec` and `readstreamspec`
`openstreamspec`	revert, resolve, shelve, or open for edit	`readstreamspec`
`readstreamspec`	display	

The denial levels for a stream spec in the P4Admin built-in editor are:

- `nowritestreamspec`

- `noopenstreamspec`
- `noreadstreamspec`

You can also work with these access levels in P4Admin as text, where:

`=writestreamspec` replaces `nowritestreamspec`

`=openstreamspec` replaces `noopenstreamspec`

`=readstreamspec` replaces `noreadstreamspec`

Before you limit or remove an access level, make sure this access level has been provided.

If you use `=writestreamspec`, `=openstreamspec`, or `=readstreamspec`, P4Admin requires that you also use an exclusionary mapping. For example:

```
writestreamspec user maria * //... ## maria has
writestreamspec for all streams
=writestreamspec user maria * -//a2/... ## except
streams in the a2 depot
```

Suppose that, for the specified depot, you want to the user to have only `readstreamspec`:

```
writestreamspec user maria * //... ##
writestreamspec for all streams
=writestreamspec user maria * -//a2/... ## removing
writestreamspec from that depot
=openstreamspec user maria * -//a2/... ## also
removing openstreamspec from that depot
```

So, whereas granting an access level implicitly grants any implied access levels, denying an access level does not implicitly remove any other access level. All denials must be explicit.

IMPORTANT: The 2020.1 release added protections modes that are specific to stream specs. By default, these permissions can exist in the protection table, but will not be used until the dm.protects.streamspec configurable has been set to `1`. If the `dm.protects.streamspec` configurable is set to `1` and any stream spec permissions exist in the protection table, the pre-2020.1 permissions no longer apply and all users who are not admin or super require explicit stream spec permissions.

User/Group	Indicates whether this line applies to a Perforce user or group.
Name	A Helix server user name or group name; can be wildcarded.
Host	The IP address of a client host; can be wildcarded.

Folder/File	The part of the depot to which access is being granted or denied. To deny access to a depot path, preface the path with a dash (-). Exclusionary mappings apply to all access levels, regardless of the access level specified in the first field.
Comment	Optional description of a table entry. Appends a comment at the end of a line using the ## symbols. For example: `write user * 10.1.1.1 //depot/test/... ## robinson crusoe`

For details about how permissions work within Helix server, see the Authoring access chapter of the *Helix Core Server Administrator Guide*.

Configuring P4Admin preferences

To configure settings for P4Admin, select **Edit > Preferences** (Windows) or **P4Admin > Preferences** (Mac). The **Preferences** dialog includes the following configuration pages:

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

Connections

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

When the application launches:

- **Restore all previously opened connections**: Do not prompt for connection settings; reconnect to the server to which you were connected during your last session.

Opening and closing connections:

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

Server Data

You can configure how much data P4Admin processes during a session to minimize server load for frequently run commands and large data transfers. The following settings are available:

- **Check server for updates every *n* minutes**: Specifies how often P4Admin checks Helix server for updated file information. Frequent checks enable P4Admin to display current file status but increase the workload on Helix server.
- **Maximum size of file to preview (excludes audio and video files)**: Limits the size of image files displayed in the Preview tab on the Files pane, to limit the amount of image data sent from Helix server to P4V.

Logging

You can configure the following logging options:

Log pane options:

- **Show p4 reporting commands**: Specifies whether the log pane in the Administration Tool window displays all commands issued by the Administration Tool, including commands issued by the Administration Tool to obtain status information from the Helix server.

Logging to a file:

- **Enable logging to file**: Logs Administration Tool activity to the specified file.
  - **Name**: Specifies the name and location of the log file.
  - **Size**: Specifies the maximum size of the log file.

Display

You can configure the following Administration Tool display and localization options:

Application:

- **Dates:** Sets the date format used throughout the Administration Tool.
  - **OS format**: Use the format that your operating system uses.
  - **Perforce standard (yyyy/mm/dd hh:mm:ss)**: Use the application standard.
- **Scale application icons based on (requires restart):** Select **Calculated primary monitor resolution** if you mainly view P4V on your primary monitor. If you view P4V on a secondary monitor with a different resolution than the primary monitor, select **Custom size** and move the slider to the scaling that is right for your secondary monitor.
- **Application color scheme (requires restart):** Select **Light theme** (default) for the classic P4V colors. Select **Dark theme** if you prefer light font on a dark background.

Localization:

- **Language used for application menus, labels, and dialogs (requires restart):** Select language.
- **Set encoding for all connections to:** Sets the character encoding for connections to a unicode-mode Helix server.

  If you do not set the encoding here, you are prompted to enter the character encoding every time you set up a connection to a unicode-mode Helix server. The encoding that you set here does not affect server connections whose character encoding has already been set at connection. If you are unsure which setting to choose, consult your Helix server administrator.

Files

You can configure the way the Administration Tool displays files and file icons:

- **Show Perforce filetype for files in the Depot tree:** Toggles display of file type in the tree pane.
- **Show revision information for files in the Depot tree:** Toggles display of revision numbers in the tree pane.

Application Font

You can configure the font family, style, and size for the application font and file content font that P4Admin uses. For file content, you can also select to **Show fixed sized fonts only**. Selecting this option limits the values available in the **Font family** list to fixed-sized fonts.

Behavior

You can configure the following Administration Tool user interface behaviors:

Drag and drop:

- **on a file, do a diff comparisons:** When selected, P4Admin launches the **Diff** dialog when you drop a file on another file.
- **anywhere within a changelist, move open files to new changelist:** When selected, P4Admin moves any open files to a new changelist when you drop a file within a changelist.
- **on a file, do nothing:** When selected, P4Admin does not do anything when you drop a file on another file.

Tools

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

Revision Graph:

- **Limit Revision Graph to ancestors and descendants**: Limits a file's integration history to ancestors and descendants (default). This option has the smallest footprint and ensures optimized performance.
- **Show Full Revision history in Revision Graph:** Displays the full integration history of the branch. With this option, the revision graph might take longer to display.

Time-Lapse View:

- **By default Time-lapse View should show**: Specifies whether Time-lapse View displays the integration/merge history for the selected file by default. You can also toggle the display of integration history in Time-lapse View.

Editor

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

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

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

> **Note**
> Any application that you've used to open a file from the context menu in the Administration Tool appears by default as an associated application on the Editor page in the Administration Tool Preferences dialog, unless you remove it.

Diff

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

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

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

To assign diff applications by file type:

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

   Enter `%1` for the name of the first file and `%2` for the name of the second file. The Administration Tool replaces these placeholders with the actual filenames when calling the diff application.
5. Click **Save**.

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

Applets

You can enable a Helix server to serve applets that can run in P4Admin. For these applets to run in P4Admin, you must enable applets in the P4Admin preferences. For more information about Perforce applets, see the *Helix Core Javascript API for Visual Tools Developer Guide*.

To enable applets to run in P4Admin:

1. Select **Allow Perforce applets to run in P4Admin.**
2. Specify the Helix server (one or more) from which you are willing to accept Perforce applets:
   - Enter the Helix server name or **`host:port`** in the **Server** field.
   - Click **Add.**
3. Click **Advanced** to specify the following settings:
   - **Save cookies from applets** (Click **Remove Cookies** to delete all applet-generated cookies)
   - **Allow applets to store data locally**
   - **Manually configure web proxy used by applets for internet access**: Enter the web proxy address and port.

Glossary

A

access level

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

admin access

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

APC

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

archive

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

atomic change transaction

Grouping operations affecting a number of files in a single transaction. If all operations in the transaction succeed, all the files are updated. If any operation in the transaction fails, none of the files are updated.

avatar

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

B

base

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

binary file type

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

branch

(noun) A set of related files that exist at a specific location in the Perforce depot as a result of being copied to that location, as opposed to being added to that location. A group of related files is often referred to as a codeline. (verb) To create a codeline by copying another codeline with the 'p4 integrate', 'p4 copy', or 'p4 populate' command.

branch form

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

branch mapping

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

branch view

A specification of the branching relationship between two codelines in the depot. Each branch view has a unique name and defines how files are mapped from the originating codeline to the target codeline. This is the same as branch mapping.

broker

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

C

change review

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

changelist

A list of files, their version numbers, the changes made to the files, and a description of the changes made. A changelist is the basic unit of versioned work in Helix server. The changes specified in the changelist are not stored in the depot until the changelist is submitted to the depot. See also atomic change transaction and changelist number.

changelist form

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

changelist number

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

check in

To submit a file to the Helix server depot.

check out

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

checkpoint

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

classic depot

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

client form

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

client name

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

client root

The topmost (root) directory of a client workspace. If two or more client workspaces are located on one machine, they should not share a client root directory.

client side

The right-hand side of a mapping within a client view, specifying where the corresponding depot files are located in the client workspace.

client workspace

Directories on your machine where you work on file revisions that are managed by Helix server. By default, this name is set to the name of the machine on which your client workspace is located, but it can be overridden. Client workspaces, labels, and branch specifications cannot share the same name.

code review

A process in Helix Swarm by which other developers can see your code, provide feedback, and approve or reject your changes.

codeline

A set of files that evolve collectively. One codeline can be branched from another, allowing each set of files to evolve separately.

comment

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

commit server

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

conflict

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

copy up

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

counter

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

CSRF

Cross-Site Request Forgery, a form of web-based attack that exploits the trust that a site has in a user's web browser.

D

default changelist

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

deleted file

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

delta

The differences between two files.

depot

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

depot root

The topmost (root) directory for a depot.

depot side

The left side of any client view mapping, specifying the location of files in a depot.

depot syntax

Helix server syntax for specifying the location of files in the depot. Depot syntax begins with: //depot/

diff

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

donor file

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

E

edge server

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

exclusionary access

A permission that denies access to the specified files.

exclusionary mapping

A view mapping that excludes specific files or directories.

extension

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

F

file conflict

In a three-way file merge, a situation in which two revisions of a file differ from each other and from their base file. Also, an attempt to submit a file that is not an edit of the head revision of the file in the depot, which typically occurs when another user opens the file for edit after you have opened the file for edit.

file pattern

Helix server command line syntax that enables you to specify files using wildcards.

file repository

The master copy of all files, which is shared by all users. In Helix server, this is called the depot.

file revision

A specific version of a file within the depot. Each revision is assigned a number, in sequence. Any revision can be accessed in the depot by its revision number, preceded by a pound sign (#), for example testfile#3.

file tree

All the subdirectories and files under a given root directory.

file type

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

fix

A job that has been closed in a changelist.

form

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

forwarding replica

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

G

Git Fusion

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

graph depot

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

group

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

H

have list

The list of file revisions currently in the client workspace.

head revision

The most recent revision of a file within the depot. Because file revisions are numbered sequentially, this revision is the highest-numbered revision of that file.

heartbeat

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

Helix server

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

Helix TeamHub

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

Helix4Git

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

hybrid workspace

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

I

iconv

A PHP extension that performs character set conversion, and is an interface to the GNU libiconv library.

integrate

To compare two sets of files (for example, two codeline branches) and determine which changes in one set apply to the other, determine if the changes have already been propagated, and propagate any outstanding changes from one set to another.

J

job

A user-defined unit of work tracked by Helix server. The job template determines what information is tracked. The template can be modified by the Helix server system administrator. A job describes work to be done, such as a bug fix. Associating a job with a changelist records which changes fixed the bug.

job daemon

A program that checks the Helix server machine daily to determine if any jobs are open. If so, the daemon sends an email message to interested users, informing them the number of jobs in each category, the severity of each job, and more.

job specification

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

job view

A syntax used for searching Helix server jobs.

journal

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

journal rotation

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

journaling

The process of recording changes made to the Helix server's metadata.

L

label

A named list of user-specified file revisions.

label view

The view that specifies which filenames in the depot can be stored in a particular label.

lazy copy

A method used by Helix server to make internal copies of files without duplicating file content in the depot. A lazy copy points to the original versioned file (depot file). Lazy copies minimize the consumption of disk space by storing references to the original file instead of copies of the file.

license file

A file that ensures that the number of Helix server users on your site does not exceed the number for which you have paid.

list access

A protection level that enables you to run reporting commands but prevents access to the contents of files.

local depot

Any depot located on the currently specified Helix server.

local syntax

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

lock

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

log

Error output from the Helix server. To specify a log file, set the P4LOG environment variable or use the p4d -L flag when starting the service.

M

mapping

A single line in a view, consisting of a left side and a right side that specify the correspondences between files in the depot and files in a client, label, or branch. See also workspace view, branch view, and label view.

MDS checksum

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

merge

1. To create new files from existing files, preserving their ancestry (branching). 2. To propagate changes from one set of files to another. 3. The process of combining the contents of two conflicting file revisions into a single file, typically using a merge tool like P4Merge.

merge file

A file generated by the Helix server from two conflicting file revisions.

metadata

The data stored by the Helix server that describes the files in the depot, the current state of client workspaces, protections, users, labels, and branches. Metadata is stored in the Perforce database and is separate from the archive files that users submit.

modification time or modtime

The time a file was last changed.

MPM

Multi-Processing Module, a component of the Apache web server that is responsible for binding to network ports, accepting requests, and dispatch operations to handle the request.

N

nonexistent revision

A completely empty revision of any file. Syncing to a nonexistent revision of a file removes it from your workspace. An empty file revision created by deleting a file and the #none revision specifier are examples of nonexistent file revisions.

numbered changelist

A pending changelist to which Helix server has assigned a number.

O

opened file

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

stream depot

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

submit

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

super access

An access level that gives the user permission to run every Helix server command, including commands that set protections, install triggers, or shut down the service for maintenance.

symlink file type

A Helix server file type assigned to symbolic links. On platforms that do not support symbolic links, symlink files appear as small text files.

sync

To copy a file revision (or set of file revisions) from the Helix server depot to a client workspace.

T

target file

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

text file type

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

theirs

The revision in the depot with which the client file (your file) is merged when you resolve a file conflict. When you are working with branched files, theirs is the donor file.

three-way merge

The process of combining three file revisions. During a three-way merge, you can identify where conflicting changes have occurred and specify how you want to resolve the conflicts.

trigger

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

two-way merge

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

typemap

A table in Helix server in which you assign file types to files.

U

user

The identifier that Helix server uses to determine who is performing an operation.

V

versioned file

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

view

A description of the relationship between two sets of files. See workspace view, label view, branch view.

W

wildcard

A special character used to match other characters in strings. The following wildcards are available in Helix server: * matches anything except a slash; ... matches anything including slashes; %%0 through %%9 is used for parameter substitution in views.

workspace

See client workspace.

workspace view

A set of mappings that specifies the correspondence between file locations in the depot and the client workspace.

write access

A protection level that enables you to run commands that alter the contents of files in the depot. Write access includes read and list accesses.

X

XSS

Cross-Site Scripting, a form of web-based attack that injects malicious code into a user's web browser.

Y

yours

The edited version of a file in your client workspace when you resolve a file. Also, the target file when you integrate a branched file.

License statements

For complete licensing information pertaining to P4V, the Helix Visual Client, P4Admin, P4Merge, and P4VJS, see the license file at https://www.perforce.com/perforce/doc.current/user/p4v_license.txt.